Avaliação de ferramentas que utilizam Processamento de Linguagem Natural aplicada ao reconhecimento de voz e texto de palavras escritas e faladas erroneamente, conforme a norma culta da língua portuguesa

Maria Clara de A. Silva[1]

[1]Universidade Federal do Ceará (UFC)
Russas – CE – Brasil

mclara.engsoftware@gmail.com

Abstract. *This meta-article describes tests and evaluations from these in google API tools in Python (voice), python NLTK library (text), IBM Watson Text to Speech service (voice), and software developed with the Quasar Speech Framework (voice). Also, an application was developed for speech recognition, whose implementation was made in Javascript language, using the Speech API. Based on the results, it was found that the google API is more efficient in identifying mispronounced words, according to the well-known Portuguese language standard.*

Resumo. *Este meta-artigo descreve testes e avaliações a partir destes nas ferramentas API da google em Python (voz), biblioteca NLTK do python (texto), serviço Text to Speech da IBM Watson (voz) e um software desenvolvido com o Framework Quasar Speech (voz). Ainda, uma aplicação foi desenvolvida para reconhecimento de fala, cuja implementação foi feita em linguagem Javascript, usando a API Speech. Com base nos resultados, foi identificado que a API da google possui mais eficiência na identificação de palavras pronunciadas erradas, conforme a norma culta da língua portuguesa.*

descreve o estilo a ser usado na confecção de artigos e resumos de artigos para publicação nos anais das conferências organizadas pela SBC. É solicitada a escrita de resumo e abstract apenas para os artigos escritos em português. Artigos em inglês deverão apresentar apenas abstract. Nos dois casos, o autor deve tomar cuidado para que o resumo (e o abstract) não ultrapassem 10 linhas cada, sendo que ambos devem estar na primeira página do artigo.

1. Introdução

O Brasil, com base nos dados levantados pelo Instituto Brasileiro de Geografia e Estatística - IBGE - em 2018, ainda possuía cerca de 11,5 milhões de pessoas

impossibilitadas de escrever, ler e interpretar, àqueles considerados “analfabetos” (SOUZA, 2015), sendo eles representados por 7% da população com mais de 15 anos que não recebeu a educação adequada na idade regular.

Como alternativa para amenizar esse problema, foram desenvolvidos alguns softwares destinados a esse público, visando auxiliá-los durante o aprendizado, dentre os quais se destaca o *Luz do saber EJA*, cuja metodologia se baseia no Método Paulo Freire de Alfabetização.

O método Paulo Freire de alfabetização é baseado nos saberes adquiridos ao longo da vida de cada educando que, contrariamente ao material didático pré-elaborado para a alfabetização, alicerçado por assuntos que muitas vezes não fazem sentido algum na vida dele, contendo frases como: “O bode viu uma bota” ou “A bota do bode”, propõe trabalhar com as “palavras geradoras”, cujo conteúdo é relacionado com o contexto em que vive o alfabetizando, de forma que ele consiga concatenar o que se aprende ao que já conhece no mundo real, podendo gerar, assim, mais interesse por parte dele nos estudos e um aprendizado mais eficaz (BECK, 2016).

No software *Luz do saber EJA* essa metodologia é abordada por meio de atividades que envolvem a entrada de dados do aluno em forma de textos, as quais, na maioria das vezes, precisam ser executadas mediante o auxílio de uma pessoa letrada, haja visto que o educando, na sua condição de analfabeto, possui determinada incapacidade de participar de uma comunicação escrita.

Com o intuito de dar mais independência às pessoas que utilizam esse tipo de aplicação, é sugerível a inserção, no *Luz do saber EJA*, de um módulo que empregue Processamento de Linguagem Natural - PLN -, ou o desenvolvimento de um novo sistema que implemente os recursos dessa subárea da Inteligência Artificial responsável por estudar a capacidade e as limitações de uma máquina em entender a linguagem dos seres humanos (BARBOSA et al., 2017).

Para tanto, este trabalho possui avaliações realizadas mediante a testes em ferramentas já existentes de reconhecimento de voz e tratamento de texto, utilizando como entrada exemplos de frases com palavras corriqueiramente pronunciadas de forma errada pelas pessoas em condição de analfabetismo, conforme o que rege a norma culta da língua portuguesa.

2. Procedimentos metodológicos

As ferramentas utilizadas nos testes apresentados aqui são API da *google* em *Python* (voz), biblioteca NLTK do *python* (texto), serviço Text to Speech da IBM Watson (voz) e um software desenvolvido com o *Framework Quasar Speech* (voz). Ainda, uma aplicação foi desenvolvida para reconhecimento de fala, cuja implementação foi feita em linguagem *Javascript*, usando a API *Speech*.

Todos os recursos de voz foram testados com as mesmas frases, nas quais as seguintes palavras foram abordadas: “Pranto” (planto), “Mi” (milho), “Cuié” (colher), “Prártico” (plástico), “Bicicreta” (Bicicleta), “Biciqueta” (Bicicleta), “Fofi” (fósforo) e “Fórfuro” (fósforo). Cada experimento foi feito com a repetição três vezes consecutivas

de cada frase, com tonalidade e velocidade semelhante de voz e no mesmo ambiente, evitando sons externos ao teste. As frases estão listadas abaixo:

Frase 1: "Eu pranto mi."
Frase 2: "A minha cuié é de prártico."
Frase 3: "Eu ando de bibicreta."
Frase 4: "Eu ando de biciqueta."
Frase 5: "Eu acendi o fogo com um palito de fofi."
Frase 5: "Eu acendi o fogo com um palito de fórfuro."

2.1. Teste da API da *google* em *Python*

A API da *google* em *Python* tem suporte para mais de 80 idiomas e trabalha basicamente com qualquer tipo de aplicação. Porém, o seu reconhecimento não acontece em tempo real, quando um comando é recebido, o sistema se conecta a um servidor online para reconhecê-lo, o que acarreta a espera de alguns segundos.

Para a realização do seu teste foi utilizada a biblioteca *Speech Recognition* e a *PyAudio*, da qual a primeira possui dependência. No código, é possível utilizar as funções que são responsáveis por ouvir e reconhecer a fala (Recogonizer()), habilitar o microfone do dispositivo que está sendo utilizado (Microphone()), reduzir o ruído (adjust_for_ambient_noise()), escolher o idioma a ser reconhecido, dentre outras.

O resultado dos testes para as frases pré-definidas são listados abaixo e o algoritmo é mostrado na **Figura 1.**

Frase 1: Saídas - **T1:** "eu planto me" **T2:** "eu planto me." **T3:** "eu planto me"

Frase 2: Saídas - **T1:** "a minha colher de plástico" **T2:** "a minha colher de plástico." **T3:** "a minha colher de plástico"

Frase 3: Saídas - **T1:** "Eu ando de bicicleta." **T2:** "Eu ando de bicicleta." **T3:** "Eu ando de bicicleta."

Frase 4: Saídas - **T1:** "Eu ando de bicicleta." **T2:** "Eu ando de bicicleta" **T3:** "Eu ando de bicicleta."

Frase 5: Saídas - **T1:** "eu acendi o fogo com um palito de fósforo" **T2:** "eu acendi o fogo com um palito de fósforo" **T3:** "eu acendi o fogo com um palito de fósforo"

Frase 6: Saídas - **T1:** "eu acendi o fogo com palito de fósforo" **T2:** "eu acendi o fogo com um palito de fósforo" **T3:** "eu acendi o fogo com palito de fósforo."

```
import speech_recognition as sr
r = sr.Recognizer()
with sr.Microphone() as s:
    r.adjust_for_ambient_noise(s)
    while True:
        audio = r.listen(s)
        print('Você disse: ', r.recognize_google(audio, language='pt'))
```

Figura 1. Algoritmo da API da *google* em *python*

Como resultado, é possível observar que todas as palavras erradas, conforme o idioma, inseridas nas frases foram reconhecidas da forma devida. Em contrapartida, a palavra que deveria ser identificada como milho, foi reconhecida como o pronome "Me", haja visto que "Mi" e "Me" possuem a mesma pronúncia e este último é validado como existente na língua portuguesa.

2.2. Avaliação da biblioteca NLTK do *python*

A Natural Language Toolkit - NLTK - é uma biblioteca da linguagem Python para Processamento de Linguagem Natural e Text Analytics, que é amplamente adotado no desenvolvimento de aplicações de PLN.

O módulo NLTK é um kit de ferramentas, cuja utilidade diz respeito a, dentre outros, separar as sentenças em um parágrafo, separar as palavras dentro de cada sentença, reconhecer padrões no texto e criar modelos de classificação que permitam, por exemplo, realizar análise de sentimentos em um conjunto de dados.

Para realizar os testes com essa biblioteca, foi necessário além de instalá-la na máquina, baixar o pacote *numpy*, com o qual pode-se trabalhar com computação científica. No código, foram usadas funcionalidades como o *sent_tokenize*, com a qual é possível reconhecer o término de frases; o método *pos_tag*, que identifica a qual classe ela pertence - verbo, substantivo, adjetivo, dentre outras -; e a função *chunk*, que identifica as entidades, como uma empresa e uma pessoa, por exemplo.

```
import nltk
texto = 'Sr. Verde matou o Coronel. Mas o Sr. Verde gosta de caju. '
frases = nltk.tokenize.sent_tokenize(texto, language='portuguese')
print(frases)

tokens = nltk.word_tokenize(texto, language='portuguese')
print(tokens)

classes = nltk.pos_tag(tokens)
print(classes)
```

Figura 2. Código do uso da biblioteca NLTK

O código implementado para a realização dos testes da biblioteca podem ser vistos na **Figura 2**. A impressão da variável "frases" obteve como resultado:

['Sr. Verde matou o Coronel.'] ['Mas o Sr. Verde gosta de caju']

O que demonstra que o algoritmo foi capaz de reconhecer "Sr." como uma abreviação, em vez do término da primeira frase, que poderia ter ocorrido devido ao ponto.

A *word.tokenize* separa o texto em palava, cada palavra é um *token*. A impressão da variável "tokens" obteve como resultado:

['Sr.', 'Verde', 'matou', 'o', 'Coronel.', 'Mas', 'o', 'Sr.', 'Verde', 'gosta', 'de', 'caju']

Por último, com a impressão da variável "classes", pôde ser visto o acerto na definição de todas as classes das palavras identificadas, como demonstrado abaixo:

[('Sr.', NNP), ('Verde', NNP), ('matou', VBD), ('Coronel.', NNP), ('Mas', CC), ('Sr.', NNP), ('Verde', JJ), ('gosta', MD), ('de', EM), ('caju', NNS)]

Os significados de cada abreviação podem ser verificados neste link https://www.clips.uantwerpen.be/pages/mbsp-tags.

Um ponto negativo que se observou ao utilizar essa ferramenta é o fato de, caso seja transcrita uma palavra com inicial maiúscula, ele reconhece como substantivo próprio, mesmo que não seja. Um exemplo disso é a classificação da palavra "Coronel", que foi considerada NNP (substantivo próprio), em vez de NN (substantivo).

2.3. Avaliação serviço *Text to Speech* da IBM *Watson*

O teste desta ferramenta foi realizado utilizando a DEMO disponível no site da empresa IBM (https://www.ibm.com/watson/br-pt/). Lá, não houve acesso a códigos-fonte, apenas à interface gráfica. Os resultados podem ser visualizados abaixo:

Frase 1: Saídas - **T1:** "Eu prole" **T2:** "Eu planto me." **T3:** "E o pranto me."

Frase 2: Saídas - **T1:** "A minha um colher de prático." **T2:** "A minha é de prática" **T3:** "Minha mãe é é de prático"

Frase 3: Saídas - **T1:** "Eu ando de bicicleta." **T2:** "Eu ando de bicicleta." **T3:** "Eu ando de bicicleta."

Frase 4: Saídas - **T1:** "Eu ando de bicicleta." **T2:** "Eu ando de bicicleta" **T3:** "Eu ando de bicicleta."

Frase 5: Saídas - **T1:** "Eu acende o fogo com um palito de fósforo." **T2:** "Erro acende o fogo conta palito diz fox." **T3:** "Os e o assédio fogo com um palito de forças"

Frase 6: Saídas - **T1:** "Eu acendi o fogo com um palito de fósforo." **T2:** "Eu acende o fogo conta alito disposto." **T3:** "Os e o assédio fogo com um palito de forças."

Ao realizar a análise, é possível observar que na **frase 1**, apenas um teste obteve o resultado esperado - interpretação de “Pranto” como “Planto” -, além de que, no **T3** nem mesmo a palavra “Eu” foi reconhecida, considerada como “E o”. Na **frase 2**, o reconhecedor considerou “Prártico” como sendo a palavra existente no idioma, “Prático”; além de não reconhecer em dois dos testes a palavra “cuié” como nenhuma existente na língua portuguesa. Em contrapartida, os substantivos “Plástico” e “Bicicleta” foram reconhecidos, apesar de, na estrutura das frases que continham essas palavras, outras terem sido interpretadas da forma errada.

2.4. Avaliação do software desenvolvido com o *Framework Quasar Speech*

O teste desta ferramenta foi realizado utilizando a DEMO disponível no site https://quasarspeechapi.surge.sh/#/. Lá, não houve acesso a códigos-fonte, apenas à interface gráfica. No entanto, a própria plataforma do *Quasar* disponibiliza tutoriais de como implementar os métodos de reconhecimento de voz. Os resultados podem ser visualizados abaixo:

Frase 1: Saídas - **T1:** “ eu planto me” **T2:** “ eu planto me” **T3:** “ eu planto me”

Frase 2: Saídas - **T1:** “ a minha colher é de plástico” **T2:** “a minha colher de plástico” **T3:** “a minha colher é de plástico”

Frase 3: Saídas - **T1:** “eu ando de bicicleta” **T2:** “eu ando de bicicleta.” **T3:** “Eu ando de bicicleta.”

Frase 4: Saídas - **T1:** “eu ando de bicicleta.” **T2:** “eu ando de bicicleta” **T3:** “eu ando de bicicleta.”

Frase 5: Saídas - **T1:** “eu acendi o fogo com palito de fósforo.” **T2:** “ eu acendi o fogo com palito de fósforo” **T3:** “ eu acendi o fogo com palito de fósforo”

Frase 6: Saídas - **T1:** “ eu acendi o fogo com palito de fósforo” **T2:** “ eu acendi o fogo com palito de fósforo” **T3:** “ eu acendi o fogo com palito de fósforo”

Ao realizar a análise, é possível observar que para todas as frases foram obtidas boas saídas, com exceção do fato de que a palavra que deveria ser identificada como milho, foi reconhecida como o pronome “Me”, haja visto que “Mi” e “Me” possuem a mesma pronúncia e este último é validado como existente na língua portuguesa. Além disso, foram ocultadas, em alguns resultados, a monossílaba “um”, o que, apesar de deixar a comunicação melhor em diálogos humanos reais, não são tão relevantes na interpretação de um contexto.

2.5. Desenvolvimento e avaliação da aplicação implementada em *Javascript*, usando a API *Speech*

A API *Speech* permite que os desenvolvedores forneçam a entrada de voz e recursos de saída de texto-para-voz em um navegador web. A própria API é independente da implementação de reconhecimento e síntese de fala subjacente e pode suportar reconhecimento e síntese baseados em servidor e em cliente / incorporado.

Para a construção dessa aplicação foi utilizado, além da linguagem *Javascript* e a API *Speech*, o *framework* Bootstrap para desenvolvimento da interface gráfica. O código da aplicação pode ser visualizado abaixo, com alguns comentários escritos na cor azul:

```
window.addEventListener('DOMContentLoaded', function(){
	var btn_gravacao = document.querySelector('#btn_gravar_audio');  //habilita o
microfone
	var transcricao_audio = ''; //Variavel responsavel por receber a transcrição
do audio
	var esta_gravando = false; //Variavel para verificação (gravando ou nao?)
	//verificação se o navegador dá suporte a API
	if(window.SpeechRecognition || window.webkitSpeechRecognition){
		var speech_api = window.SpeechRecognition ||
window.webkitSpeechRecognition;
		var recebe_audio = new speech_api();
		// definição de parâmetros de controle para o uso da api
		recebe_audio.continuous = true; //continua ou nao a gravação
		recebe_audio.interimResults = true; // o resultado pode ser alterado ou
não
		recebe_audio.lang = "pt-BR"; //idioma a ser usado
		//métodos de controle  (garante a mudança de texto do botao)
		recebe_audio.onstart = function(){
			esta_gravando = true;
			btn_gravacao.innerHTML = 'Gravando/Parar gravação';
		};
		recebe_audio.onend = function(){
			esta_gravando = false;
			btn_gravacao.innerHTML = 'Iniciar gravação';
		};
		//método para capturar o resultado do audio
		recebe_audio.onresult = function(event){
			var interim_transcript = '';
			for(var i = event.resultIndex; i < event.results.length; i++){
				if(event.results[i].isFinal){
					transcricao_audio +=
event.results[i][0].transcript;
				}else{
					interim_transcript +=
event.results[i][0].transcript;
				}
```

```
                        var resultado = transcricao_audio || interim_transcript;
                        document.getElementById('campo_texto').innerHTML =
resultado;
                }
        };
        btn_gravacao.addEventListener('click', function(e){
                if(esta_gravando){
                        recebe_audio.stop();
                        return;
                }
                recebe_audio.start();
        }, false);
    }else{
        console.log('Navegador não apresenta suporte a api');
    }
}, false);
```

Dentre as funções que podem ser utilizadas com a API, destacam-se:

- **onstart:** Define um callback que é disparado quando o serviço de reconhecimento começou a ouvir o áudio com a intenção de reconhecer.
- **onResult:** Define um callback que é disparado quando o reconhecedor de voz retorna um resultado.
- **onerror:** Define um callback que é acionado quando ocorre um erro de reconhecimento de voz.
- **onend:** Define um callback que é disparado quando o serviço foi desligado. O evento deve sempre ser gerado quando a sessão termina, não importa o que a razão.

4. Considerações finais

Os estudos foram úteis para o conhecimento de diferentes tipos de ferramentas e poderá direcionar o estudo para a análise profunda dos algoritmos utilizados nos recursos e, com isso, poderão ser encontradas meios de melhorá-los de forma a atender ao reconhecimento de padrões relacionados a palavras pronunciadas de forma incorreta, conforme a norma culta.

Dentre as ferramentas testadas, a que demonstrou os melhores resultados foi a API da *google* em *Python*. Outros recursos também foram encontrados, como a API *Houndify* e a *Microsoft Bing Voice Recognition*, no entanto não foram testadas pelo fato de a primeira não reconhecer a língua portuguesa e a segunda ser paga.

References

SOUZA, Renata Mara de. Analfabetismo no Brasil. 2015. 35. Trabalho de Conclusão de Curso - Faculdade de Pará de Minas, Pará de Minas - MG, 2015.

BECK, C. Método Paulo Freire de alfabetização. Andragogia Brasil. Disponível em:

https://andragogiabrasil.com.br/metodo-paulo-freire-de-alfabetizacao/. Acesso em: 01 maio 2019

Bird, Steven, Edward Loper e Ewan Klein (2009), *Natural Language Processing with Python* . O'Reilly Media Inc.

BARBOSA, J. et al. Introdução ao processamento de linguagem natural usando python. III Escola Regional de Informatica do Piauí, v. 1, p. 336-360, 2017