Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Departamento de Informática e Matemática Aplicada
Bacharelado em Ciência da Computação

Geração de prosódia para o português brasileiro em sistemas *text-to-speech*

Felipe Cortez de Sá

Natal-RN
Junho de 2018

Felipe Cortez de Sá

Geração de prosódia para o português brasileiro em sistemas *text-to-speech*

Monografia de Graduação apresentada ao Departamento de Informática e Matemática Aplicada do Centro de Ciências Exatas e da Terra da Universidade Federal do Rio Grande do Norte como requisito parcial para a obtenção do grau de bacharel em Ciência da Computação.

Orientador

Dr. Carlos Augusto Prolo

Universidade Federal do Rio Grande do Norte – UFRN
Departamento de Informática e Matemática Aplicada – DIMAp

Natal-RN

Junho de 2018

Monografia de Graduação sob o título *Geração de prosódia para o português brasileiro em sistemas* text-to-speech apresentada por Felipe Cortez de Sá e aceita pelo Departamento de Informática e Matemática Aplicada do Centro de Ciências Exatas e da Terra da Universidade Federal do Rio Grande do Norte, sendo aprovada por todos os membros da banca examinadora abaixo especificada:

Dr. Carlos Augusto Prolo
Orientador
Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte

Dr. Antônio Carlos Gay Thomé
Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte

Dra. Erica Reviglio Iliovitz
Departamento de Letras
Universidade Federal do Rio Grande do Norte

Natal-RN, 20 de junho de 2018.

Dedicado a mim

Agradecimentos

Agradeço à minha família. A Antônia. A Marc. A Prolo, pela orientação. A todos os meus professores.

Some few people are born without any sense of time. As consequence, their sense of place becomes heightened to an excruciating degree. They lie in tall grass and are questioned by poets and painters from all over the world. These time-deaf are beseeched to describe the precise placement of trees in the spring, the shape of snow on the Alps, the angle of sun on a church, the position of rivers, the location of moss, the pattern of birds in a flock. Yet the time-deaf are unable to speak what they know. For speech needs a sequence of words, spoken in time.

Alan Lightman, Einstein's Dreams

Geração de prosódia para o português brasileiro em sistemas *text-to-speech*

Autor: Felipe Cortez de Sá
Orientador(a): Dr. Carlos Augusto Prolo

Resumo

Com a cada vez mais forte presença de smartphones e home assistants no cotidiano, grandes empresas de tecnologia vêm desenvolvendo sistemas de conversação baseados em fala, denominadas voice user interfaces. Apesar dos avanços, é perceptível que os sistemas de síntese de voz, especialmente para o português brasileiro, deixam a desejar quanto à naturalidade da fala gerada. Um dos fatores principais que contribuem para isso é a prosódia, isto é, entonação, ritmo e acento da fala. Este trabalho investiga sistemas text-to-speech existentes através do estudo de seus algoritmos para síntese de voz e geração de prosódia para diversas línguas, com foco no português brasileiro. São explicitados os desafios encontrados, é feito um levantamento de modelos de análise prosódica na linguística e propõem-se possíveis soluções para tornar a geração de voz mais próxima à humana.

Palavras-chave: text-to-speech, prosódia, voice user interfaces

Prosody generation for Brazilian Portuguese in text-to-speech systems

Author: Felipe Cortez de Sá
Advisor: Carlos Augusto Prolo, Ph.D.

Abstract

With the evergrowing presence of smartphones and home assistants in our daily lives, technology companies have been developing two-way conversation systems, that is, voice user interfaces. Despite its recent improvements, text-to-speech programs still sound artificial, especially for their Brazilian Portuguese voices. A big contributing factor for that is the lack of accurate prosody, that is, pitch, length and emphasis. This thesis explores existing text-to-speech systems, especially those for which there are Brazilian Portuguese voices, focusing on their prosody generation modules. We highlight challenges of prosody generation, review prosodic analysis in the Linguistics field and propose possible solutions for improving text-to-speech quality.

Keywords: text-to-speech, prosody, voice user interfaces

Lista de figuras

Lista de tabelas

Lista de Códigos

Lista de abreviaturas e siglas

TTS – *text-to-speech*

INTSINT – *International Transcription System for Intonation*

HMM – *Hidden Markov Model*

SSML – *Speech Synthesis Markup Language*

XML – *eXtensible Markup Language*

W3C – *World Wide Web Consortium*

MBRPSOLA – *Multi-Band Resynthesis Pitch Synchronous Overlap and Add*

API – *Application Programming Interface*

JSON – *JavaScript Object Notation*

REST – *Representational State Transfer*

AJAX – *Asynchronous JavaScript And XML*

Lista de símbolos

ms (milissegundos)

Hz (Hertz)

Sumário

1 Introdução

Interfaces humano-computador que utilizam a voz, denominadas *voice user interfaces*, antigamente vistas apenas na ficção científica, hoje são uma realidade e estão disponíveis em *smartphones* e ambientes *desktop*. De acordo com (DUTOIT, 1997; JURAFSKY; MARTIN, 2009), há uma grande área de aplicação para essas interfaces, como a acessibilidade, permitindo que deficientes visuais possam ouvir texto sem a necessidade de gravação prévia de seu conteúdo, ferramenta de ensino de linguagens e auxílio à pesquisa na linguística. Além disso, com o aumento da popularidade de sistemas embarcados, é importante investigar novas formas de interação humano-máquina, e a síntese de fala, juntamente com o reconhecimento, permitem comunicação de duas vias com esses sistemas.

Os serviços mais populares e robustos que temos atualmente são implementações proprietárias de grandes empresas, como Siri (Apple Inc., 2011), Cortana (Microsoft Corp., 2014) e Alexa (Amazon.com, Inc., 2014). Apesar da praticidade e ganho de acessibilidade providos por essas interfaces, os serviços disponíveis sintetizam voz com resultados perceptivelmente artificiais, principalmente para a língua portuguesa. Uma das causas da artificialidade é a prosódia empregada, isto é, o ritmo, entonação e acento.

1.1 Objetivos

Este trabalho tem como objetivo geral propor melhorias para o módulo de prosódia afetiva para sistemas TTS com suporte ao português brasileiro através de uma revisão da área da fonologia e estudo de estado da arte de sistemas TTS, com atenção especial a como a prosódia é abordada.

1.2 Organização do trabalho

No capítulo 2, é feita uma breve explanação dos conceitos abordados neste trabalho, com foco em TTS e prosódia.

No capítulo 3, é feita uma revisão da literatura, fazendo um levantamento dos sistemas *text-to-speech* existentes tanto para o inglês quanto para o português brasileiro e como a prosódia é modelada em cada um deles. Mostramos como trabalhos recentes abordam síntese de fala. Também são descritos os trabalhos existentes em análise de prosódia em um contexto não necessariamente computacional.

No capítulo 4, é descrito o sistema desenvolvido, justificando a abordagem com base na revisão da literatura. Explica-se a implementação do software, descrevendo sua arquitetura, as linguagens de programações utilizadas e o funcionamento.

No capítulo 5, apresentamos os resultados obtidos com a implementação.

2 Fundamentação teórica

2.1 Prosódia

2.1.1 Componentes

(CAGLIARI et al., 1981, p. 150) cita. Stress (loudness and phonatory force) Syllabic length F0, intensidade, duração Intonational tune Sintagma entoacional Além desses componentes principais, Taylor (2009) descreve downdrift e microprosódia.

2.1.2 Função prosódica

Taylor (2009) divide a prosódia em três tipos distintos:

- Afetiva: prosódia pura, emoção, estado mental e atitude.
- Suprassegmental: quando uma mensagem é dita de maneira declarativa, sem conteúdo afetivo significante, dá o nome de *discourse neutral*. utiliza um modelo de prosódia em que parte supra-segmental não é conteúdo prosódico verdadeiro, mas sim um outro aspecto do componente verbal.
- Aumentativa: assegura a comunicação efetiva de uma mensagem.

2.1.3 Prosódia como elemento extra-textual

Taylor (2009) argumenta que a geração de prosódia afetiva é árdua e depende de uma compreensão do texto. Ainda fala que, até a data de publicação do livro, nenhuma solução satisfatória foi encontrada e os sistemas TTS atuais pecam nesse aspecto.

2.1.4 Contornos melódicos

Moraes (2008) analisa para a mesma frase "Renata jogava" utilizando o modelo ToBI uma mesma frase falada com intenções diferentes analisando como a entoação afeta a intenção percebida.

2.2 Sistemas TTS

Burnett e Shuang (2010) definem *text-to-speech* como "o processo de geração automática de fala a partir de texto ou texto anotado" (tradução nossa). Sistemas TTS são compostos por múltiplos subsistemas, como mostramos a seguir.

2.2.1 Estrutura

Na literatura, encontramos diversas arquiteturas específicas de um sistema TTS. Dutoit (1997) propõe um diagrama geral, dividindo sistemas em duas partes principais:

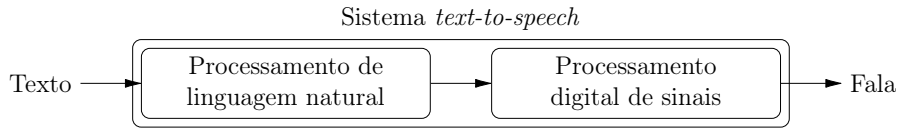

Figura 1: Arquitetura geral de sistemas TTS (adaptado de Dutoit (1997))

É comum encontrar em outros trabalhos o termo *front end* para o bloco de processamento de linguagem natural e *back end* para o bloco de processamento digital de sinais. Doravante utilizaremos essa nomenclatura.

2.2.2 *Front end*

As definições a seguir são adaptadas de (JURAFSKY; MARTIN, 2009):

2.2.2.1 Normalização de texto

O processo de conversão de texto para fala começa com o processamento do texto a fim de gerar uma representação grafêmica.

A primeira etapa da normalização é a separação em sintagmas, isto é, encontrar os limites de cada frase do texto.

Em seguida, devem-se transformar símbolos, abreviações, siglas e outras *non-standard words* em suas representações pronunciáveis. Como exemplo, "R$ 50" deve ser lido como "cinquenta reais".

Por último, deve ser realizada a desambiguação de homônimos heterófonos: em "Gosto de pão", "gosto" pode ser pronunciada como "gôsto" ou "gósto".

2.2.2.2 Conversão grafema-fone

Palavras não-padrão são colocadas num dicionário de pronúncia. O resto é calculado de acordo com regras letra-som. Context-dependent e independent. Abordagens por machine learning ou regras. A conversão grafema-fone (ou fonema) consiste em transformar o texto normalizado em fones, ou seja, uma sequência de caracteres em uma sequência de fones.

2.2.2.3 Geração de prosódia

A partir do texto e fones gerados nas etapas anteriores, deve-se estimar uma curva F0, ritmo e ênfase.

2.2.3 Back end

Com o texto de entrada transformado em fones e informação prosódica, um *back end* é responsável por gerar uma forma de onda. Jurafsky e Martin (2009) separam algoritmos de síntese em três paradigmas: síntese concatenativa, síntese por formantes e síntese articulatória.

- Síntese articulatória
- Síntese por formantes
- Síntese concatenativa Síntese concatenativa pode trabalhar com diferentes atomicidades: nível de palavra, difonos e fones individuais.

Abordagem utilizada pelo programa MBROLA. Consiste em gravar fala, separar pedaços de dois em dois.

Algumas vozes para o MaryTTS (SCHRöDER; TROUVAIN, 2003) utilizam HMMs, isto é, Modelos ocultos de Markov para *unit selection*. Há, inclusive, uma voz brasileira feita a partir de HMMs: (COUTO et al., 2010). Projetos mais recentes como (WU; WATTS; KING,

2016; ZE; SENIOR; SCHUSTER, 2013) utilizam redes neurais para estimação de parâmetros. O trabalho da Google informa? que a estimação da curva F0 é uma possível melhoria.

3 Revisão da literatura

3.1 Sistemas TTS para o português brasileiro

Destacamos aqui sistemas TTS desenvolvidos por ordem cronológica a fim de evidenciar a evolução das tecnologias utilizadas e as tendências para trabalhos futuros.

Aiuruetê (BARBOSA et al., 1999) Usa PSOLA pra recodar unidades armazenadas (ortofones). Síntese concatenativa.

eSpeakNG (DUNN, 2006) Síntese por formantes (com consoantes gravadas) ou concatenativa utilizando o *back end* MBROLA. É possível utilizar apenas o *front end* passando *flags* para a execução do programa.

Microsoft (BRAGA et al., 2008) Baseado em HMM. A stochastic-based LTS (Letter-to-Sound) converter to predict phonetic transcriptions for out-of-vocabulary words and the prosody models, which are data- driven using a prosody tagged corpus of 2000 sentences. Síntese concatenativa.

(COUTO et al., 2010) Apesar de não ter sido oficialmente integrado à ferramenta, foi desenvolvido com base no *framework* MaryTTS um sistema completo para português brasileiro baseado em HMMs. O projeto também acarretou no desenvolvimento do programa de conversão grafema-fone LaPS-G2P (BARBOSA et al., 2003). Foi estendido por (COSTA et al., 2012) para funcionar de maneira *stand-alone*, isto é, ser utilizado sem necessidade de instalação do *framework* MaryTTS.

LianeTTS (SERPRO, 2011) Projeto da SERPRO LianeTTS. Utiliza MBROLA como *back end*. A geração de prosódia é simples, baseada em tabela. Síntese concatenativa.

LINSE-DNN (MAIA; SEARA, 2017) Descreve melhorias para o *back-end* baseado em redes neurais profundas. Síntese concatenativa? Ou por formantes?

MBROLA MBROLA é uma ferramenta para geração de voz baseada em síntese por difonos desenvolvida com o objetivo de fomentar pesquisas acadêmicas em geração de prosódia (DUTOIT et al., 1996). É utilizada como *back-end* para diversos sistemas TTS, como MaryTTS, e possui três vozes disponíveis para o português brasileiro.

```
_ 150 50 150
t 70 50 125
e 125 50 75
c 70 50 125
e 125 50 75
c 70 50 125
e 116 20 232 80 300
_ 150 50 150
```

Código 3.1: Exemplo de arquivo de entrada para MBROLA

3.1.0.1 Formato

Em cada linha, tem-se um fone ou um silêncio representado pelo *underscore* seguido por uma duração em milissegundos e, por último, um ou mais pares de porcentagem e frequência em Hertz determinando alvos para a curva F0. Como exemplo, na penúltima linha temos o fone e com duração de 116 ms e dois alvos para altura, 232 Hz em 20% e 300 Hz em 80%.

Cada voz gravada provê uma tabela com os fones que podem ser utilizados. Utilizamos neste trabalho a voz br3 desenvolvida por Denis R. Costa disponível no site oficial do projeto MBROLA.

3.1.1 Modelos de análise entoacional

3.1.1.1 Teoria métrica-autossegmental

Utilizada por Moraes (2008) para analisar uma mesma frase com diferentes intenções.

3.1.1.2 IPO

Sistema proposto pela Escola Holandesa. Utilizada por (MIRANDA, 2015) para analisar a prosódia no português brasileiro.

3.1.1.3 INTSINT

Key (frequência base) Range (intervalo entre ponto mais alto e mais baixo do f0). INTSINT é um sistema de anotação para prosódia. Top (T), Mid(M), Bottom (B). Higher (H), Same (S), Lower(L). Acentos: Upstepped (S), Downstepped (D).

Utilizado por (CELESTE; REIS, 2012) para analisar entoação para o português brasileiro.

Algoritmo MOMEL: MOdelisation de MELodie.

3.1.1.4 DaTo (*Dynamic Tones*)

Apesar de o modelo ToBI para anotação entoacional ter sido utilizado para analisar o português brasileiro em diversos trabalhos, Lucente (2014) argumenta que há características perceptíveis que a notação não consegue expressar, propondo o modelo DaTo com base na entonação do português brasileiro.

3.1.2 Prosódia afetiva em sistemas TTS

3.1.2.1 SSML

A linguagem de marcação *Speech Synthesis Markup Language* foi criada motivada pela dificuldade de predição computacional de pronúncia (TAYLOR; ISARD, 1997). Quando originalmente proposta, diferentes sistemas TTS permitiam anotações extra-textuais para auxiliar a estimação de parâmetros, mas usuários tinham que aprender um sistema de anotação para cada programa diferente. A proposta da SSML é que os sistemas TTS recebam o texto anotado numa linguagem unificada. A linguagem foi adotada por soluções *open-source* como MaryTTS e espeak-ng, além dos sistemas proprietários encontrados em Alexa, Google Assistant e Cortana. A especificação é mantida pela W3C (BURNETT; SHUANG, 2010). A especificação cita os elementos *emphasis*, *break* e *prosody* como marcadores que podem auxiliar o processador de linguagem natural a gerar parâmetros prosódicos apropriados.

```
<speak>
    Siga
    <emphasis level="strong">aquele</emphasis>
    carro.
</speak>
```

Código 3.2: Exemplo de texto anotado com SSML

3.1.2.2 EmotionML

EmotionML (SCHRÖDER; BURKHARDT, 2014) foi criada para várias coisas, uma delas é ajudar algoritmos a determinarem prosódia. (CHARFUELAN; STEINER, 2013) descreve um *framework* para implementação de determinação de prosódia a partir de anotações em EmotionML utilizando o *framework* MaryTTS.

Como explicitado por (TAYLOR, 2009), não há um acordo quanto ao sistema mais apropriado para representar emoções. A linguagem de marcação tem suporte a múltiplos sistemas descritivos, como categorias, dimensões, appraisals e action tendencies. As categorias podem receber valores discretos (como pode ser visto no código 3.3), determinando se uma emoção está presente ou valores contínuos, determinando a intensidade de uma emoção específica (como pode ser visto na figura 3.4), permitindo múltiplas categorias simultaneamente.

```
<emotionml version="1.0" xmlns="http://www.w3.org/2009/10/emotionml">
  <emotion category-set="http://www.w3.org/TR/emotion-voc/xml#everyday-categories">
  <emotion>
    <category name="happy" />
    Que bom te ver!
  </emotion>
</emotionml>
```

Código 3.3: Exemplo de texto anotado com EmotionML com parâmetros discretos

```
<speak version="1.1" xmlns="http://www.w3.org/2001/10/synthesis"
        xmlns:emo="http://www.w3.org/2009/10/emotionml"
        xml:lang="en-US">
    <s>
        <emo:emotion
            category-set="http://www.w3.org/TR/emotion-voc/xml#everyday-categories">
           <emo:category name="worried" value="0.4"/>
        </emo:emotion>
        Precisa de ajuda?
    </s>
```

```
</speak>
```

Código 3.4: Exemplo de texto anotado com SSML e EmotionML (adaptado de (SCHRÖDER; BURKHARDT, 2014))

3.1.2.3 Festival

O sistema TTS festival disponibiliza uma maneira de especificar entoação seguindo o modelo ToBI.

```
(Utterance Words
 (The
  (boy ((accent L*)))
  saw
  the
  (girl ((accent H*) (tone L-)))
  with
  the
  (telescope ((accent H*) (tone H-H%))))))
```

Código 3.5: Anotações no modelo ToBI para o sistema TTS Festival

4 Editor de prosódia

4.1 Implementação

4.1.1 espeak-ng

Foi utilizado o programa *open-source* espeak-ng (DUNN, 2006) para realizar a normalização de texto e realizar a conversão grafema-fone, ou seja, obter a partir do texto de entrada uma representação em fones. A saída gerada é passada para o programa desenvolvido neste trabalho através da biblioteca `subprocess` da linguagem Python. O comando utilizado para obter os grafemas pode ser visto no código 4.1.

- A *flag* `-v` seleciona uma voz. Neste caso, `pt-br`
- `-x` e `-q` escrevem fones na saída em vez da fala sintetizada
- `-sep=/` separa fones utilizando o caractere `/`

```
$ espeak-ng -v pt-br 'Bom dia' -x -q --sep=/
$ b/'o/N dZ/'i/;/&
```

Código 4.1: Utilização do programa espeak e saída correspondente

Apesar da existência de outras ferramentas para *front-end* para o português brasileiro, optamos por esta pela facilidade de instalação, marcação de ênfase disponível, etc.

4.1.2 MBROLA

Cada voz do MBROLA tem suporte a um conjunto específico de dífonos. O programa `sampa_mbrola.py` foi desenvolvido para converter a saída gerada em fones suportados pela voz do MBROLA utilizada. O equivalente ao fone `&` gerado pelo espeak é `a` na voz do MBROLA, por exemplo. A ferramenta de conversão lê a tabela do arquivo `sampa_mbrola.tbl`, e substitui cada fone pelo equivalente corrigido. A tabela possui três colunas: o fone que

o espeak produz, o fone equivalente para o MBROLA e um exemplo numa palavra do português. Em 4.2 é possível ver algumas linhas do arquivo.

```
& | a | v_a_le
6 | @ | tam_a_nho
n | n | _n_unca
```

Código 4.2: Linhas da tabela de conversão

As durações padrão para cada fone estão contidas na tabela do arquivo `durations.tbl`, adaptado da tabela utilizada pelo LianeTTS (SERPRO, 2011). Cada linha contém o fone e uma duração em milissegundos.

4.1.3 Arquitetura

Foi utilizada uma arquitetura cliente-servidor REST (FIELDING, 2000) comumente vista em aplicações web contemporâneas. A interface gráfica consome uma API REST. A maneira como os módulos se comunicam pode ser vista na figura 2.

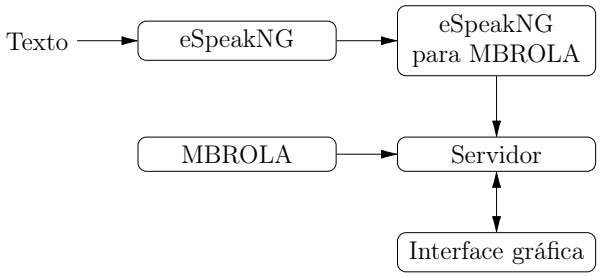

Figura 2: Arquitetura do sistema desenvolvido

4.1.4 Módulo de prosódia

O programa foi codificado em Python em sua versão 3.6. Pega resultado do espeak-ng, processa com editor gráfico e gera MBROLA.

Para alterar a prosódia manualmente, foi desenvolvido um editor gráfico para *web* utilizando HTML, CSS e JavaScript. A duração e altura de cada fone pode ser especificado arrastando barras de controle. O editor se comunica com o espeak-ng e MBROLA através de um servidor programado em Python utilizando o *framework* Flask para prover *endpoints* de uma API REST.

4.1.5 Endpoints

Foram criados dois endpoints para a API, tornando possível a comunicação entre interface gráfica e servidor.

[POST] /api/espeak Recebe um texto como entrada e gera uma resposta JSON com lista de fones

[POST] /api/mbrola Recebe uma lista de fones no formato MBROLA descrito na seção 3.1 e gera uma resposta com o nome do arquivo de áudio contendo a fala sintetizada

4.1.6 Editor gráfico

Editor gráfico utilizando o *framework* Vue.js. Comunica-se com o servidor através de *AJAX*

5 Resultados

Com o levantamento dos sistemas TTS para o português brasileiro, notamos que falta suporte à síntese expressiva proporcionada pela geração de contornos F0 relacionados a prosódia afetiva. Linguagens de marcação como SSML e EmotionML já foram integradas a *frameworks* para TTS e sistemas comerciais, mas ainda não há integração entre essas linguagens e síntese de voz para o português brasileiro.

Estudando os modelos de anotação entoacional, percebe-se que ainda não há uma solução mais adequada para analisar o português brasileiro de forma a

Com o desenvolvimento da aplicação *web* deste trabalho, dá-se um passo inicial para a integração de prosódia afetiva a sistemas TTS para o português brasileiro.

6 Considerações finais

Neste trabalho foi apresentada a definição de um sistema TTS, bem como a descrição de seus componentes principais e possíveis implementações para cada módulo. Também foi realizada uma busca dos sistemas TTS *open-source* e comerciais existentes. Com o objetivo de melhorar a geração de prosódia, foi feita uma revisão de prosódia na linguística, descrevendo os principais sistemas de anotação usados na fonologia entoacional, principalmente aplicados ao português brasileiro. Investigamos como a prosódia funciona nos sistemas encontrados e identificamos que, enquanto a geração de prosódia suprassegmental em trabalhos recentes já produz resultados satisfatórios, ainda há desafios quando à síntese de fala expressiva relacionada à prosódia afetiva e aumentativa.

6.1 Trabalhos futuros

Foi identificado que ainda há um significante desafio para a geração de prosódia afetiva automática a partir de um texto não anotado. Possíveis soluções para avanço nesta área incluem:

- Expansão do sistema desenvolvido neste trabalho para adicionar suporte a mais modelos notação entoacional.
- Avaliação estatística da qualidade de falas geradas com a aplicação desenvolvida comparada a outros sistemas *open-source* e comerciais
- Adicionar suporte a SSML e EmotionML ao sistema desenvolvido, gerando curvas F0 a partir de linguagens de marcação
- Uso técnicas de técnicas de *Natural Language Understanding* para gerar notação EmotionML automaticamente.
- Desenvolvimento de corpus anotados com prosódia para o português brasileiro.

Referências

Amazon.com, Inc. *Alexa*. 2014. Disponível em: <https://developer.amazon.com/alexa>. Acesso em 25 de setembro de 2017.

Apple Inc. *Siri*. 2011. Disponível em: <https://www.apple.com/ios/siri/>. Acesso em 25 de setembro de 2017.

BARBOSA, F. et al. Grapheme-phone transcription algorithm for a Brazilian Portuguese TTS. In: SPRINGER. *International Workshop on Computational Processing of the Portuguese Language*. [S.l.], 2003. p. 23–30.

BARBOSA, P. A. et al. Aiuruete: A high-quality concatenative text-to-speech system for brazilian portuguese with demisyllabic analysis-based units and a hierarchical model of rhythm production. In: . [S.l.: s.n.], 1999.

BRAGA, D. et al. HMM-based Brazilian Portuguese TTS. *Braga et al.(eds), Propor*, p. 47–50, 2008.

BURNETT, D.; SHUANG, Z. W. *Speech Synthesis Markup Language (SSML) Version 1.1*. [S.l.], 2010. Disponível em: <http://www.w3.org/TR/2010/REC-speech-synthesis11-20100907/>. Acesso em 5 de junho de 2018.

CAGLIARI, L. C. et al. Elementos de fonética do português brasileiro. [sn], 1981.

CELESTE, L.; REIS, C. Análise entonativa formal: Intsint aplicado ao português. *Journal of Speech*, v. 2, n. 2, p. 3–21, 2012.

CHARFUELAN, M.; STEINER, I. Expressive speech synthesis in MARY TTS using audiobook data and emotionML. In: *INTERSPEECH*. [S.l.: s.n.], 2013. p. 1564–1568.

COSTA, E. S. et al. Um sintetizador de voz baseado em hmms livre: Dando novas vozes para aplicaç oes livres no português do brasil. In: *Workshop de Software Livre*. [S.l.: s.n.], 2012.

COUTO, I. et al. An open source HMM-based text-to-speech system for Brazilian Portuguese. In: *7th international telecommunications symposium*. [S.l.: s.n.], 2010.

DUNN, R. H. *espeak-ng*. 2006. Disponível em: <https://github.com/espeak-ng/espeak-ng>. Acesso em 29 de outubro de 2017.

DUTOIT, T. *An introduction to text-to-speech synthesis*. [S.l.: s.n.], 1997. (Text, Speech and Language Technology 3).

DUTOIT, T. et al. The MBROLA project: Towards a set of high quality speech synthesizers free of use for non commercial purposes. In: IEEE. *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*. [S.l.], 1996. v. 3, p. 1393–1396.

FIELDING, R. T. Rest: architectural styles and the design of network-based software architectures. *Doctoral dissertation, University of California*, 2000.

JURAFSKY, D.; MARTIN, J. H. *Speech and Language Processing (2nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2009. ISBN 0131873210.

LUCENTE, L. Uma abordagem fonética na fonologia entoacional. *Fórum Linguístico*, v. 11, n. 1, p. 79–95, 2014.

MAIA, R.; SEARA, R. Um sistema TTS baseado em redes neurais profundas usando parâmetros síncronos de pitch. p. 388–392, 2017.

Microsoft Corp. *Cortana*. 2014. Disponível em: <https://www.microsoft.com/en-us/windows/cortana>. Acesso em 25 de setembro de 2017.

MIRANDA, L. *Análise da entoação do português do Brasil segundo o modelo IPO*. Tese (Doutorado) — Dissertação de mestrado em Língua Portuguesa. Rio de Janeiro: UFRJ, 2015.

MORAES, J. A. de. The pitch accents in Brazilian Portuguese: analysis by synthesis. In: *Proc. Speech Prosody*. [S.l.: s.n.], 2008. p. 389–397.

SCHRÖDER, M.; BURKHARDT, F. *Emotion Markup Language (EmotionML) 1.0*. [S.l.], maio 2014. Disponível em: <http://www.w3.org/TR/2014/REC-emotionml-20140522/>. Acesso em 8 de junho de 2018.

SCHRöDER, M.; TROUVAIN, J. The german text-to-speech synthesis system mary: A tool for research, development and teaching. *International Journal of Speech Technology*, Springer, v. 6, n. 4, p. 365–377, 2003.

SERPRO. *LianeTTS*. 2011. Disponível em: <http://intervox.nce.ufrj.br>. Acesso em 7 de junho de 2018.

TAYLOR, P. *Text-to-speech synthesis*. [S.l.]: Cambridge University Press, 2009.

TAYLOR, P.; ISARD, A. Ssml: A speech synthesis markup language. *Speech communication*, Elsevier, v. 21, n. 1-2, p. 123–133, 1997.

WU, Z.; WATTS, O.; KING, S. Merlin: An open source neural network speech synthesis system. *Proc. SSW, Sunnyvale, USA*, 2016.

ZE, H.; SENIOR, A.; SCHUSTER, M. Statistical parametric speech synthesis using deep neural networks. In: IEEE. *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. [S.l.], 2013. p. 7962–7966.

APÊNDICE A – Primeiro apêndice

```
import sqlite3
import sampa_mbrola
from flask import Flask, g, jsonify, render_template, abort, request
from flask_cors import CORS

app = Flask(__name__)
CORS(app)

@app.route('/api/espeak', methods=['POST'])
def frontend():
    text = request.form['text']
    converter = sampa_mbrola.Converter()
    sentence = converter.convert_sentence(text)
    return jsonify(sentence.dictify())

@app.route('/api/mbrola', methods=['GET'])
def mbrola(page):
    pass

if __name__ == '__main__':
    app.run(debug=True)
```

Código A.1: Servidor

```
import sys
import re
import subprocess
import random
import json
from collections import OrderedDict
from typing import List

def flatten(lst: list):
    return [item for sublist in lst for item in sublist]
```

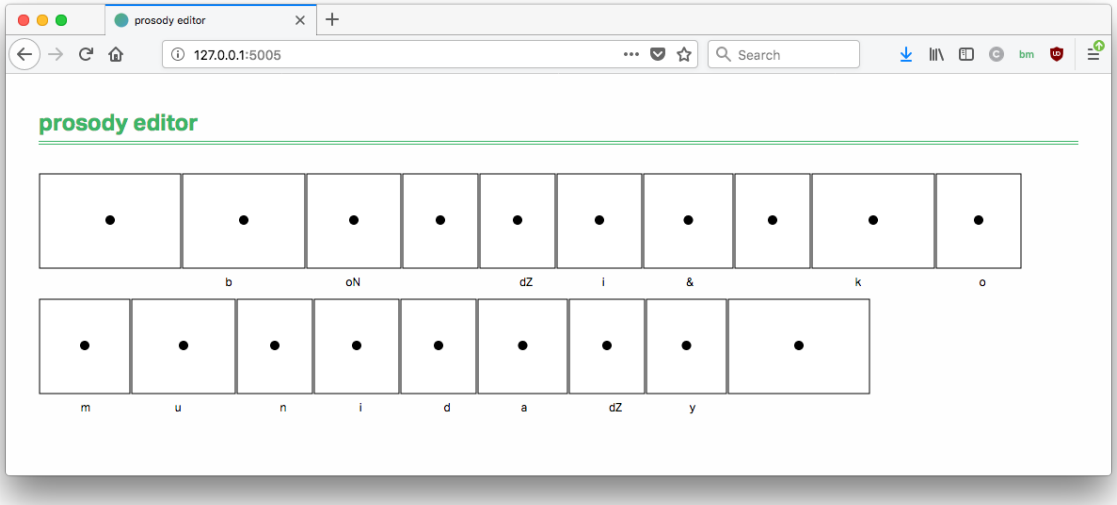

Figura 3: Editor gráfico

```
class Phone():
    def __init__(
            self,
            phone_sampa: str,
            phone_mbrola: str,
            duration: int,
            pitch_changes: list
    ):
        self.phone_sampa = phone_sampa
        self.phone_mbrola = phone_mbrola
        self.duration = duration
        self.pitch_changes = pitch_changes

    def as_line(self):
        return "{} {} {}".format(
            self.phone_mbrola,
            self.duration,
            " ".join([str(item) for item in flatten(self.pitch_changes)])
        )

class Sentence():
    def __init__(self, phones: List[Phone]=None):
        if phones is None:
            self.phones = []
        else:
            self.phones = phones
```

```
    def mbrola_phones(self):
        return [phone.phone_mbrola for phone in self.phones]

    def dictify(self):
        return [vars(phone) for phone in self.phones]

    def __repr__(self):
        return "\n".join([phone.as_line() for phone in self.phones])


class Converter():
    def __init__(self):
        self.load_sampa_mbrola()
        self.load_durations()

    def load_sampa_mbrola(self):
        equivs = {}
        with open("sampa_mbrola.tbl") as f:
            for line in f:
                k, v, _ = line.split()
                equivs[k] = v

        self.equivs = OrderedDict(
            sorted(equivs.items(), key=lambda t: -len(t[0])))

    def load_durations(self):
        durations = {}
        with open("durations.tbl") as f:
            for line in f:
                k, v = line.split()
                durations[k] = v

        self.durations = durations

    def convert_phoneme(self, sentence: str) -> tuple:
        """Returns first phone from the sentence"""

        if sentence[0] == " ":
            return ("_", 1)
        elif self.equivs:
            # s is a special case, needs to peek next
            if sentence[0] == "s":
```

```
            if sentence[1] in "aeiou&":
                return ("s", 1)

        for equiv in self.equivs.items():

            if re.match(re.escape(equiv[0]), sentence):
                # print("match:", equiv[0], phoneme, "=", equiv[1])
                return (equiv[1], len(equiv[0]))

    # print("didn't match", phoneme)
    return (phoneme[0], 1)

def get_duration(self, phoneme: str) -> int:
    if self.durations and phoneme in self.durations:
        return self.durations[phoneme]
    else:
        return 100

def convert_sentence(self, input_str: str) -> Sentence:
    sentence = Sentence()
    sentence.phones.append(Phone(" ", "_", 150, [[50, 150]]))

    ignored = ["@", "\n", ",", "'", "^", ";"]

    sampa = self.text_to_sampa(input_str)
    sampa = sampa.replace("'", "")
    print(";; ", sampa)

    while sampa:
        if sampa[0] not in ignored:
            converted = self.convert_phoneme(sampa)
            phone_sampa = sampa[:converted[1]]
            sampa = sampa[converted[1]:]

            duration = self.get_duration(converted[0])
            phone = Phone(
                phone_sampa = phone_sampa,
                phone_mbrola = converted[0],
                duration = int(duration),
                pitch_changes = [[50, 150]] # percentage, Hz
            )

            sentence.phones.append(phone)
```

```
            else:
                sampa = sampa[1:]

        sentence.phones.append(Phone(" ", "_", 150, [[50, 150]]))
        return sentence

    def text_to_sampa(self, sentence: str) -> str:
        espeak_str = "espeak-ng -v pt-br '{}' -x -q".format(sentence)

        p = subprocess.Popen(espeak_str, stdout=subprocess.PIPE, shell=True)
        (output, err) = p.communicate()
        p_status = p.wait()
        output = output.decode("utf-8")
        output = output.replace("\n", "").strip()
        return output


if __name__ == "__main__":
    converter = Converter()

    for line in sys.stdin:
        print(";;", line)
        print(converter.convert_sentence(line))
```

Código A.2: Conversor eSpeakNG-MBROLA