UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

PROCESSAMENTO DE TEXTO ESCRITO EM LINGUAGEM NATURAL PARA UM SISTEMA CONVERSOR TEXTO-FALA

THIAGO MINHAQUI OECHSLER

BLUMENAU
2009

2009/2-24

THIAGO MINHAQUI OECHSLER

PROCESSAMENTO DE TEXTO ESCRITO EM LINGUAGEM NATURAL PARA UM SISTEMA CONVERSOR TEXTO-FALA

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Profa. Joyce Martins , Mestre – Orientadora

BLUMENAU
2009

2009/2-24

PROCESSAMENTO DE TEXTO ESCRITO EM LINGUAGEM NATURAL PARA UM SISTEMA CONVERSOR TEXTO-FALA

Por

THIAGO MINHAQUI OECHSLER

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Profa. Joyce Martins, Mestre – Orientadora, FURB
Membro:	Prof. Mauro Marcelo Mattos, Doutor – FURB
Membro:	Prof. José Roque Voltolini da Silva – FURB

Blumenau, 10 de dezembro de 2009.

Dedico este trabalho a todas as pessoas com deficiência visual, que poderão se beneficiar com os resultados do mesmo. Meu pensamento esteve sempre voltado a vocês.

AGRADECIMENTOS

À minha família, que me deu inspiração e condições para realizar tudo isto. Em especial à minha mãe, que sugeriu o tema deste trabalho, com o qual me identifiquei e gostei.

À minha namorada, Raquel, que me auxiliou nas questões da língua portuguesa, me sugeriu melhorias ao programa e me obrigou a trabalhar no projeto durante todos os finais de semana, sacrificando o único tempo que posso ficar ao seu lado.

À minha orientadora, Joyce Martins, por ter me aceito como aluno orientando, por toda a paciência em entender minhas idéias e pelo incentivo de fazer o meu melhor.

A *Faculté Polytechnique de Mons* (Bélgica), pelo projeto desenvolvido, que norteou minhas idéias e solidificou os resultados do trabalho.

Ao brasileiro Denis R. Costa, criador dos bancos de fonemas. Sem este, não haveria como colocar à prova a síntese das palavras.

A ACEVALI, e em especial a Luana, João e a diretora Vera, que avaliaram o projeto com honestidade, e me indicaram novos caminhos e estudos a realizar sobre o assunto.

Na atividade revolucionária, o transformar a si mesmo coincide com o transformar as circunstâncias.

Karl Marx

RESUMO

Este trabalho apresenta a especificação e a implementação de um protótipo de sistema texto-fala para a língua portuguesa, com a capacidade de processar um vocabulário irrestrito, realizando o pré-processamento léxico e sintático do texto, o tratamento de abreviaturas e, por fim, a síntese (pronúncia) de palavras, números cardinais e siglas, na ordem que são apresentadas no texto, com entonação prosódica. Para o processamento acústico foi utilizado o sintetizador MBROLA.

Palavras-chave: Sistema texto-fala. Processamento de linguagem natural. Síntese de texto. Análise linguística. Processamento prosódico.

ABSTRACT

This work presents the specification and implementation of a text-to-speech prototype system, for the portuguese language, with the ability of processing unrestricted vocabulary, performing lexical and syntactic text pre-processing, word abbreviation expanding, and ultimately the synthesis of the words, cardinal numbers, and abbreviations, in text appearance order with prosodic intonation. For acoustic processing it was used MBROLA synthesizer.

Key-words: Text-to-speech system. Natural language processing. Text synthesizing. Linguistic analysis. Prosodic processing.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

ACEVALI - Associação de Cegos do Vale do Itajaí

BPL – *Borland Project Library*

DSC – Departamento de Sistemas e Computação

FURB – Universidade Regional de Blumenau

GALS – Gerador de Analisadores Léxico e Sintático

IA – Inteligência Artificial

RF – Requisito Funcional

RNF – Requisito Não-Funcional

SMS - *Short Message Service*

TBB – Transcrição Biunívoca Brasileira

UML – *Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

No filme 2001 Uma Odisseia no Espaço, o computador central da espaçonave, nomeado HAL, fala com os tripulantes da nave, obedecendo a ordens e dando instruções. O filme de 1968 já tratava da possibilidade de um computador emitir sons em forma de fala e comunicar-se na língua do seu operador. Esta facilidade na operação de um computador representada no filme é, nos dias de hoje, tema de estudos e de investimentos na procura por facilitar a interação entre usuário e computador. Para operar um computador comum atualmente é necessário que a pessoa seja alfabetizada, saiba usar *mouse* e teclado, e, principalmente, não tenha deficiência visual severa (não seja cego). Estas condições se tornam ainda mais essenciais quando evidencia-se o uso da escrita como veículo de troca de informação entre homem-máquina. Por sua vez, desde o lançamento do filme até hoje, a computação já apresenta algumas soluções para estas necessidades, como por exemplo, sistemas de conversão texto-fala (*Text-to-speech*) ou síntese da fala. Este tipo de software, difundido principalmente na língua inglesa, pode auxiliar pessoas com deficiência visual, e até mesmo analfabetos, a usarem um computador. Alguns destes sistemas já podem ser encontrados gratuitamente na internet, inclusive para o sistema operacional Windows. Porém, na língua portuguesa, em comparação à língua inglesa, não existem muitos sistemas de conversão texto-fala, o que evidencia uma necessidade importante de estudar esta questão. Com um sistema como este, é possível que uma pessoa que não saiba ler e escrever possa apontar o cursor do *mouse* sobre um texto em uma página de internet e ouvir o conteúdo da mesma. Da mesma forma, um deficiente visual poderia utilizar as setas de navegação do teclado, para que o sistema percorresse os textos da página, sintetizando-os em voz. Estes dois exemplos demonstram o quanto um sistema conversão texto-fala pode contribuir para massificação da informática e a inclusão digital.

Diante do exposto, neste trabalho é proposto o desenvolvimento de um sistema para conversão texto-fala a partir do processamento de texto escrito em linguagem natural. Segundo Simões (1999, p. 13), “pode se definir um sistema de síntese de fala como sendo um sistema capaz de produzir sinais de fala de maneira artificial”. A síntese de voz normalmente é composta de um conjunto de etapas que envolvem desde a identificação do contexto a ser sintetizado até a geração das ondas da fala. Uma etapa intermediária é a conversão do texto para unidades linguísticas equivalentes aos sons da linguagem. O problema é que esta relação não é direta e varia de um idioma para outro, exigindo assim que sejam implementadas regras

para que o sistema de síntese possa efetuar essa etapa do processo (FRANZEN, 2002, p. 13). De forma geral, as etapas do processo de conversão texto-fala são: pré-processamento, análise linguística, transcrição fonética, processamento prosódico e síntese.

Com exceção da última etapa (síntese), as primeiras dedicam-se a realizar o processamento das palavras. As estratégias usadas na implementação destas etapas têm impacto direto no desempenho final do sistema e podem comprometer o resultado desejado – uma fala inteligível e natural. Isso denota a importância destas quatro primeiras etapas e é exatamente o que é investigado neste trabalho, ficando excluída a etapa de processamento acústico (síntese), em que de fato é realizada a síntese da fala.

1.1 OBJETIVOS DO TRABALHO

Este trabalho tem como objetivo desenvolver um protótipo para converter uma entrada em linguagem natural para uma linguagem com gramática formal.

Os objetivos específicos podem ser listados da seguinte maneira:

a) processar adequadamente textos escritos em português;
b) processar um vocabulário restrito;
c) definir regras de processamento computacional correspondentes às normativas da gramática e reforma ortográfica da língua portuguesa;
d) gerar um arquivo texto contendo o código intermediário em linguagem formal, que permita a validação do resultado do programa, sem a necessidade de um módulo para síntese da fala;
e) criar biblioteca de funções para permitir que projetos de extensão futuros possam implementar um módulo de processamento acústico.

1.2 ESTRUTURA DO TRABALHO

Apresentados a introdução e os objetivos, no capítulo dois é descrita a fundamentação teórica para o trabalho. Nele é explicado o que são sistemas de conversão texto-fala e as etapas para o processamento textual. Ainda no capítulo dois são explanadas algumas

características gramaticais da língua portuguesa e trabalhos correlatos.

O capítulo três apresenta a especificação, a implementação e o funcionamento do sistema de conversão texto-fala proposto. Esse capítulo trata isoladamente cada etapa implementada, regras e estratégias desenvolvidas para o tratamento do texto, algoritmos e técnicas utilizadas e os resultados do processamento.

O capítulo quatro discorre sobre as conclusões provenientes do desenvolvimento desse trabalho, bem como as possíveis extensões do mesmo.

2 FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica foi organizada de acordo com as etapas que são necessárias para o desenvolvimento do protótipo proposto. Tem-se que a primeira seção deste capítulo traz uma breve explicação sobre os sistemas de conversão texto-fala. A segunda seção discorre sobre o estudo gramatical da língua portuguesa, em especial o estudo da fonética. Da terceira até a sexta seção são apresentadas as etapas de um sistema texto-fala, quais sejam: pré-processamento do texto, analisador linguístico, identificador fonético e processador prosódico. Por fim, a sétima seção apresenta trabalhos correlatos.

2.1 SISTEMAS DE CONVERSÃO TEXTO-FALA

Os primeiros sistemas para produção artificial da fala surgiram a partir de 1922, data do mais antigo sintetizador de formantes elétrico que se tem registro (GOMES, 1998, p. 5). O mecanismo fundamental de um sistema de conversão texto-fala consiste em receber um texto de entrada e efetuar a síntese do texto (falar), e esta característica fundamental é que difere estes sistemas de outras máquinas falantes como gravadores ou equipamentos que produzem sons a partir da concatenação de palavras ou sentenças isoladas com vocabulários limitados (DUTOIT, 1996 apud FRANZEN, 2002, p. 15).

Os progressos recentes em síntese de fala têm produzido sistemas de conversão texto-fala com boa inteligibilidade, mas a qualidade do som e a naturalidade ainda continuam apresentando um desempenho baixo (OSTERMANN FILHO, 2002, p. 36). Entretanto, Ostermann Filho (2002, p. 36) afirma que “a qualidade dos produtos tem alcançado um nível adequado para várias aplicações, tais como multimídia e telecomunicações. Com alguma informação audiovisual, ou animação facial, é possível aumentar a inteligibilidade da voz, consideravelmente”.

2.2 GRAMÁTICA NORMATIVA DA LÍNGUA PORTUGUESA

A linguagem natural, apesar de ser uma forma irrestrita de comunicação, também segue normas e regras. “A história, o registro e a sistematização dos fatos de uma língua constituem matéria da Gramática” (CEGALLA, 1985, p.16). Os sistemas de conversão texto-fala devem compreender as normas da língua para conquistarem um bom desempenho na sua finalidade.

> A gramática normativa enfoca a língua como é falada em determinada fase de sua evolução: faz o registro sistemático dos fatos linguísticos e dos meios de expressão, aponta normas para a correta utilização oral e escrita do idioma, em suma, ensina a falar e escrever a língua-padrão corretamente. (CEGALLA, 1985, p. 16).

Dentro do domínio da gramática, o estudo da síntese da fala para a língua portuguesa deve abranger quatro das cinco partes do estudo do idioma: a fonética, a morfologia, a sintaxe e a semântica. “A fonética é a parte da gramática que estuda os vários sons ou fonemas linguísticos” (ALMEIDA, 1969, p. 24). A morfologia ocupa-se das diversas classes das palavras, isoladamente, analisando-lhes a estrutura, a formação, as flexões e propriedades. Almeida (1969, p. 24) completa afirmando que “a morfologia é a parte que estuda a palavra em si, quer no elemento material, isto é, quanto à forma, quer no elemento imaterial, ou seja, quanto a idéia que ela encerra”. A sintaxe é a parte que estuda a palavra não em si, mas com relação às outras que com ela se unem para exprimir o pensamento. A sintaxe estuda a frase, quer completa quer incompleta (ALMEIDA, 1969, p. 24). Já a semântica, “a última parte, correspondente a estilística, visa o lado estético e emocional da atividade linguística, em oposição ao aspecto intelectivo, científico das outras” (CEGALLA, 1985, p. 17).

Para um sistema conversor texto-fala fica claro que a parte mais importante do estudo da gramática está na fonética. É a fonética que trata de questões importantes como: fonemas, vogais, semivogais e consoantes, divisão silábica, encontros vocálicos e consonantais.

2.2.1 Fonemas

Fonemas[1] são as menores unidades sonoras da fala. São os sons elementares e distintivos que articulados e combinados, formam as sílabas, os vocábulos e a teia da frase na comunicação oral. Ao pronunciar a palavra *aflito*, por exemplo, são emitidas três sílabas (a - fli - to) e seis fonemas (/a/ - /f/ - /l/ - /i/ - /t/ - /o/). Pode-se perceber então que em uma sílaba pode haver um ou mais fonemas (CEGALLA, 1997, p. 21).

O ideal seria que cada fonema correspondesse a uma só letra e vice-versa, mas isso não acontece. É que o sistema ortográfico da língua portuguesa não é rigorosamente fonético, mas ainda está preso à origem etimológica das palavras. Escreve-se, por exemplo, *exame*, em vez de *ezame*, porque este substantivo vem da palavra latina *examen* (CEGALLA, 1997, p. 21).

Os fonemas da língua portuguesa classificam-se:

a) vogais: são fonemas sonoros ou sons laríngeos, que chegam livremente ao exterior sem fazer ruído;
b) semivogais: são os fonemas /i/ e /u/ átonos que se unem a uma vogal, formando com esta uma só sílaba, tal como em *va**i**, ande**i**, o**u**ro* ou *ág**u**a*;
c) consoantes: são ruídos provenientes da resistência que os órgãos bucais opõem à corrente de ar.

"A vogal é o elemento básico, suficiente e indispensável para a formação da sílaba. As consoantes e as semivogais são fonemas dependentes: só podem formar a sílaba com o concurso das vogais" (CEGALLA, 1997, p. 24). O português utiliza 34 fonemas, sendo treze vogais, dezenove consoantes e duas semivogais, que estão representados no quadro 1.

1 Manosso (2008) afirma que ainda não existe uma Transcrição Biunívoca Brasileira (TBB) aceita como padrão para os estudos lingüísticos. Isto é, não existe um padrão para escrever os fonemas do idioma nacional. Dessa forma, nesse trabalho é adotada a proposta de transcrição encontrada na obra de Manosso (2008), onde os fonemas são escritos entre barras. Também é feita a seguinte padronização: palavras são escritas em itálico, letras em itálico e negrito e frases entre aspas. Assim, tem-se o seguinte exemplo: na frase "Bom dia, senhor!", o fonema /r2/ corresponde à letra ***r*** da palavra *senho**r***.

	fonema[1]	características fonéticas	exemplos[2]
vogais	/á/	aberta, frontal, oral, não arredondada	***á**tomo, **a**rte*
	/â/	semi-aberta, central, oral, não arredondada	*p**a**no, r**a**mo, l**a**nho*
	/ã/	semi-aberta, central, nasal, não arredondada	***an**tes, **am**plo, maç**ã**, **âm**bito*
	/é/	semi-aberta, frontal, oral, não arredondada	*m**é**trica, p**e**ça*
	/ê/	semi-fechada, frontal, oral, não arredondada	*m**e**do, p**ê**ssego*
	/ẽ/	semi-fechada, frontal, nasal, não arredondada	*s**em**pre, **êm**bolo, c**en**tro, conc**ên**trico, t**êm**, tamb**ém**[3]*
	/ó/	semi-aberta, posterior, oral, arredondada	***ó**tima, **o**va*
	/ô/	semi-fechada, posterior, oral, arredondada	*r**o**lha, av**ô***
	/õ/	semi-fechada, posterior, nasal, arredondada	***om**bro, **on**tem, c**ôm**puto, c**ôn**sul*
	/i/	fechada, frontal, oral, não arredondada	***i**tem, silv**í**cola*
	/ĩ/	fechada, frontal, nasal, não arredondada	*s**im**ples, s**ím**bolo, t**in**ta, s**ín**crono*
	/u/	fechada, posterior, oral, arredondada	***u**va, **ú**tero*
	/ũ/	fechada, posterior, nasal, arredondada	*alg**um**, pl**úm**beo, n**un**ca, m**u**ito*
consoantes	/m/	nasal, sonora, bilabial	***m**arca*
	/n/	nasal, sonora, alveolar	***n**ervo*
	/ñ/	nasal, sonora, palatal	*arra**nh**ado*
	/b/	oral, oclusiva, bilabial, sonora	***b**arco*
	/p/	oral, oclusiva, bilabial, surda	***p**ato*
	/d/	oral, oclusiva, linguodental, sonora	***d**ata*
	/t/	oral, oclusiva, linguodental, surda	***t**elha*
	/g/	oral, oclusiva, velar, sonora	***g**ato*
	/k/	oral, oclusiva, velar, surda	***c**arro, **q**uanto*
	/v/	oral, fricativa, labiodental, sonora	***v**ento*
	/f/	oral, fricativa, labiodental, surda	***f**arelo*
	/z/	oral, fricativa, alveolar, sonora	***z**ero, ca**s**a, e**x**alar*
	/s/	oral, fricativa, alveolar, surda	***s**eta, **c**ebola, e**s**pe**ss**o, e**xc**e**ss**o, a**ç**úcar, au**x**ílio, a**sc**eta*
	/j/	oral, fricativa, pós-alveolar, sonora	***g**elo, **j**arro*
	/x/	oral, fricativa, pós-alveolar, surda	***x**arope, **ch**uva*
	/r/	oral, vibrante, sonora, uvular	***r**ato, ca**rr**oça*
	/r/	oral, vibrante, sonora, alveolar	*va**r**iação*
	/ʎ/	oral, lateral aproximante, sonora, palatal	*cava**lh**eiro*
	/l/	oral, lateral aproximante, sonora, alveolar	***l**uz*
semivogais	/y/	oral, palatal, sonora	*u**i**vo, mã**e**, á**r**ea, t**êm**, tamb**ém**[3]*
	/w/	oral, velar, sonora	*a**u**tomático, móve**l**, pã**o**, fala**m**[4]*

[1] Foi utilizado um conjunto de grafemas adaptado à realidade brasileira, que não corresponde integralmente ao Alfabeto Fonético Internacional.
[2] Em ortografia oficial do português.
[3] Os grafemas em negrito nas palavras *têm* e *também* representam o encontro vocálico da vogal /ẽ/ com a semivogal /y/.
[4] Os grafemas em negrito na palavra *falam* representam o encontro vocálico da vogal /ã/ com a semivogal /w/.

Fonte: adaptado de MANOSSO (2008).

Quadro 1 – Fonemas da língua portuguesa

A definição da lista de fonemas do português tem uma dose de arbitrariedade.

Atualmente o dicionário da língua portuguesa incorpora alguns estrangeirismos como no caso das palavras *hardware* e *hub* que apresentam o /h/ aspirado. Diante disso, seria de se esperar que esse fonema fosse incluído na lista dos fonemas do português. No entanto, seu uso está restrito a palavras estrangeiras incorporadas ao idioma (MANOSSO, 2008).

"A conclusão a que se chega é que os fonemas que podem ser considerados genuínos do idioma são aqueles de uso amplo e não repelidos pelos hábitos fonéticos da comunidade" e pelas normas cultas (MANOSSO, 2008).

2.2.2 Divisão silábica

Na língua portuguesa, a sílaba se forma necessariamente com uma vogal, a que se agregam, ou não, semivogais ou consoantes. A divisão das sílabas faz-se pela silabação, isto é, pronunciando as palavras por sílabas. São os hábitos fonéticos que instituem grande influência sobre a divisão silábica. "A divisão de qualquer vocábulo deve-se mais à soletração do que propriamente aos elementos constitutivos segundo a etimologia" (BARBOSA, 1973, p. 24). Desta forma, existem as seguintes normas para divisão silábica:

> 1) A consoante inicial não seguida de vogal permanece na sílaba que a segue [...].
> 2) No meio do vocábulo a consoante não seguida de vogal é sempre conservada na sílaba que a precede [...].
> 3) Não se separam os elementos dos encontros consonânticos iniciais de sílaba, nem dos dígrafos [...].
> 4) O **sc** medial biparte-se, ficando o *s* numa sílaba e o **c** na sílaba imediata [...].
> 5) O **s** dos prefixos **bis**, **cis**, **des**, ***trans*** e o **x** de prefixo **ex** vão para a sílaba imediata quando esta se inicia por vogal [...].
> 6) As vogais idênticas e as consoantes geminadas **cc**, **cç**, **rr** e **ss** separam-se, ficando uma na sílaba que as precede e outra na sílaba seguinte.
> 7) Não se separam as vogais dos ditongos – crescentes e decrescentes – nem as dos tritongos [...]. (BARBOSA, 1973, p. 25).

2.3 PRÉ-PROCESSAMENTO DO TEXTO

Para tratar um texto escrito em português, um sistema de conversão texto-fala deve primeiramente eliminar sentenças estranhas e formatar o texto de acordo com o vocabulário suportado.

> Além dos símbolos ortográficos do alfabeto latino (eventualmente acompanhados de sinais diacríticos ou cedilha), um texto genérico em português pode apresentar uma grande variedade de caracteres, tais como sinais de pontuação, aspas, algarismos

> diversos e símbolos especiais [...]. Mesmo os símbolos do alfabeto latino podem ser usados em configurações particulares que impossibilitam a sua leitura direta; este é o caso por exemplo, de abreviações, cuja leitura exige a sua expansão em um ou mais termos que devem ser de conhecimento prévio do leitor. (GOMES, 1998, p. 15).

Com isso, o pré-processamento do texto faz o isolamento e o tratamento inicial das palavras, formando-se uma lista de palavras do texto de entrada[2]. É também nesta etapa que são substituídas as abreviações, as siglas e os símbolos especiais por suas respectivas formas extensas. Os algarismos também podem receber tratamento, sendo convertidos para sua forma por extenso, mas com pronúncia literal, não entrando no mérito de sua função na frase. Ainda nesta etapa, deve ser efetuada uma análise sintática[3] do texto, para verificar se as palavras encontradas pertencem à linguagem definida.

2.4 ANALISADOR LINGUÍSTICO

O analisador linguístico define as classes gramaticais das palavras encontradas no texto de entrada. É a partir do resultado desta classificação que o programa irá definir a pronúncia de cada palavra. Portanto, não é incorreto afirmar que o resultado do analisador linguístico está diretamente relacionado à naturalidade e a inteligibilidade da síntese final da fala. Para isto, o analisador lingüístico pode utilizar duas técnicas: a classificação por comparação e a classificação por regras posicionais.

A classificação por comparação é relativamente simples. Trata-se de obter as palavras do texto (já separadas pelo pré-processador de texto) e compará-las com uma tabela que contém os principais termos do universo determinado (pronomes, preposições, conjunções, advérbios, artigos, interjeições, substantivos, adjetivos, verbos e algumas das conjugações dos verbos irregulares mais comuns). Porém, a língua portuguesa apresenta algumas palavras que podem exercer funções sintáticas distintas. “Diversas palavras presentes na tabela de classificação gramatical podem desempenhar duas ou mais funções gramaticais, conforme o contexto que são utilizadas” (GOMES, 1998, p. 24). Com isso, a técnica de comparação não se mostra suficiente para determinar classificação correta.

---

[2] A primeira fase para o tratamento sistemático de um texto em linguagem natural pode ser feita por um analisador léxico, que consiste em identificar sequências de caracteres que constituem as palavras (REBELO, 2002, p. 6).

[3] A lista de palavras reconhecida pelo analisador léxico é usada por um analisador sintático, que verifica se as mesmas se encontram na sequência correta, de acordo com a estrutura gramatical, baseada na própria gramática normativa do idioma.

A classificação por regras posicionais apresenta um grau de dificuldade de implementação mais elevado do que técnica anterior, em que a palavra recebe sua classificação com base na classificação das palavras vizinhas através de um conjunto de regras. A definição destas regras está diretamente relacionada com a função do sistema conversor texto-fala. Uma estratégia eficiente é dividir o classificador em dois momentos: no primeiro momento este deve definir e atribuir a classe gramatical nas palavras que possuam mais de uma classificação na tabela, e num segundo momento, deve atribuir uma classe para as palavras que não receberam nenhuma classificação (GOMES, 1998, p. 26).

2.5 IDENTIFICADOR FONÉTICO

A transcrição ortográfico-fonética consiste em fazer a transformação da sequência ortográfica em uma cadeia de símbolos que represente a sequência de sons que compõe cada uma das palavras do texto (SIMÕES, 1999, p. 52). O identificador fonético pode executar a transcrição-fonética utilizando-se de uma técnica por regras (a exemplo do analisador lingüístico).

> A técnica de implementação mais simples é a transcrição por regras, que procura deduzir os fones a serem pronunciados com base no contexto em que se insere cada letra do texto de entrada. Para línguas como o português, na qual a correspondência entre letras e fones é razoavelmente estável, esta técnica permite obter bons resultados; para o inglês, ao contrário, a sua eficiência é menor, pois a pronúncia associada a letras ou grupos de letras pode sofrer variações fortes e muitas não sistemáticas [...]. (GOMES, 1998, p. 30).

É também durante a etapa de transcrição fonética que se faz uso de um dicionário de exceções. Este dicionário é uma base de dados que lista todas as palavras transcritas incorretamente pelo sistema juntamente com sua transcrição apropriada.

Outra função desta etapa é identificar a sílaba tônica das palavras, também muito importante para a naturalidade do resultado final. A língua portuguesa apresenta algumas características que facilitam a identificação da sílaba tônica. Toda sílaba apresenta uma vogal como núcleo, podendo estar cercada por consoantes ou outras vogais (desempenhando um papel de semivogal). No português existem três possibilidades para o posicionamento da sílaba tônica lexical: ao final da palavra (palavra oxítona), imediatamente antes da última sílaba (palavra paroxítona) e imediatamente antes da penúltima sílaba (palavra proparoxítona) (CEGALLA, 1985, p. 20). Palavras que contenham acento agudo ou circunflexo têm a

posição da sílaba tônica automaticamente determinada. Como as palavras proparoxítonas são sempre acentuadas, a determinação da sílaba tônica é trivial. Finalmente, em palavras não acentuadas, a sílaba tônica é determinada através de um conjunto de regras levando em conta a terminação da palavra, tomada a partir da primeira vogal da última sílaba. A identificação da sílaba tônica é uma das etapas que mais será afetada com a nova reforma ortográfica, já que a mesma traz novas regras para palavras paroxítonas, palavras que diferenciam pares e outros casos, que antes tornavam esta tarefa trivial.

2.6 PROCESSADOR PROSÓDICO

"O estudo especial da acentuação denomina-se – *prosódia*. Resulta o acento da íntima associação de certas qualidades físicas dos sons da fala, tais como: a *intensidade* [...]; a *altura* [...]; o *timbre* [...]; e a *quantidade* [...]" (LIMA, 1991, p. 24). O estudo da prosódia pode ser aplicado em cada palavra, em um grupo de palavras ou para toda uma frase. De forma geral, os grupos prosódicos quase sempre coincidem com a identificação de uma oração no texto. Porém, segundo Gomes (1998, p. 37), "em certos casos, a definição de grupo prosódico como equivalente ao conceito de oração torna-se inadequada". Por isso, é importante que o processador prosódico adote uma estratégia mais refinada do que simplesmente identificar orações. Frases que não contêm verbos, por exemplo, não constituem uma oração, porém devem ser identificadas como grupo prosódico.

Esta classificação em grupos prosódicos irá determinar características como entonação (freqüência fundamental), duração do segmento e intensidade sonora. O processador prosódico também é responsável por determinar as durações segmentais dos termos transcritos, definindo grupos interrogativos, imperativos e afirmativos/subjuntivos.

2.7 TRABALHOS CORRELATOS

É possível encontrar exemplos de sistema de conversão texto-fala derivados de trabalhos acadêmicos e desenvolvidos principalmente para a língua inglesa. No âmbito nacional são encontrados desde trabalhos de conclusão de curso a dissertações de mestrado,

dentre os quais foram selecionados três trabalhos, sendo eles: o sistema de conversão texto-fala de Gomes (1998), o protótipo de um sintetizador de voz descrito por Bento (1995) e o protótipo desenvolvido por Hubner (1992).

2.7.1 Sistema de conversão texto-fala para a língua portuguesa utilizando a abordagem de síntese por regras

Empregando algumas regras elaboradas pelo próprio autor (GOMES, 1998) para efetuar a conversão do texto em fala, o trabalho apresenta uma solução completa de síntese da fala para qualquer texto escrito em português. Ele apresenta uma descrição genérica dos sistemas de conversão texto-fala, incluindo um breve histórico e comentários sobre a estrutura e operação do sistema apresentado.

Também trata de cada uma das diferentes etapas necessárias para o processamento do texto, que, no trabalho em questão, se apresentam divididos em: pré-processador, classificador gramatical, divisor silábico, transcritor ortográfico-fonético, processador prosódico e, por fim, processamento dos sinais de fala. Este último implementa o sintetizador por formantes de Klatt (KLATT, 1980), modelo cientificamente comprovado para a construção de aplicações de síntese da fala.

2.7.2 Protótipo de um sintetizador de voz utilizando redes neurais

Utilizando-se da idéia de redes neurais, mais especificamente a topologia *Backpropagation*, o protótipo foi desenvolvido por Bento (1995) na linguagem C++, depois de ser constatado que o ambiente Turbo Pascal (inicialmente escolhido) não apresentou as características necessárias para o desenvolvimento do mesmo. Conforme a evolução do trabalho, são estipuladas várias topologias para o uso das redes neurais, que vão sendo aprimoradas até que se tenha um modelo definitivo que é utilizado para treinar a rede e, consequentemente, processar o texto. O trabalho não define um universo determinado da linguagem natural e, neste ponto, se propôs a processar qualquer texto em português.

O trabalho não apresenta a etapa de síntese da fala por ter tido dificuldade em

encontrar uma biblioteca de fonemas[4], que é utilizada para emitir os sinais acústicos. Porém, o trabalho descreve como se daria o processo de reprodução do som em posse de uma biblioteca.

2.7.3 Interface em linguagem natural

O principal objetivo deste projeto foi, segundo Hubner (1992, p. 5), utilizar-se de conceitos da Inteligência Artificial (IA) aliada a uma linguagem de programação que utiliza o mecanismo de inferência (o Prolog), para converter uma entrada em linguagem natural em uma linguagem formal. O protótipo criado foi limitado a um vocabulário de universo específico da administração financeira, o ciclo econômico financeiro. Utilizando um dicionário de palavras, o texto era analisado pelo protótipo e, então, o programa tomava determinadas ações de acordo com o texto analisado.

A utilização do Prolog, segundo Hubner (1992, p. 7), é uma estratégia muito interessante, pois é uma linguagem muito eficaz para resolução de lógicas de problemas, tais como os proporcionados pelo processamento de linguagem natural. O projeto em questão teve a preocupação de gerar funções genéricas para possibilitar a extensão do programa. Contudo, a programação em Prolog requer um conhecimento específico e uma habilidade de abstração diferenciada e não linear.

[4] Assume-se como biblioteca de fonemas um banco de sons correspondentes aos fonemas da linguagem natural.

3 DESENVOLVIMENTO DO SOFTWARE

Neste capítulo são apresentados o desenvolvimento do protótipo proposto e sua utilização, sendo que nas seções seguintes tem-se:

a) a especificação dos Requisitos Funcionais (RF) e dos Requisitos Não-Funcionais (RNF);
b) a lista de fonemas suportados pelo software;
c) o formato do arquivo de entrada;
d) os diagramas de casos de uso, de classes e de seqüência;
e) o formato do arquivo de saída;
f) a implementação do pré-processamento do texto, do analisador lingüístico, do identificador fonético e do processador prosódico.

Também são detalhadas técnicas e ferramentas utilizadas na implementação, assim como as dificuldades encontradas e os resultados obtidos.

3.1 REQUISITOS DO SOFTWARE A SER DESENVOLVIDO

O protótipo do sistema de conversão texto-fala deve:

a) disponibilizar uma interface para entrada de texto em português (RF);
b) disponibilizar uma interface para entrada de dados para dicionários (RF);
c) criar um arquivo resultante do processo de conversão, contendo código intermediário fundamental para síntese da fala (RF);
d) ser transparente durante o processo de transcrição, demonstrando os resultados obtidos com o processamento do texto (RF);
e) utilizar arquivos texto para armazenamento de dados inerentes ao processo de conversão (RNF);
f) efetuar o processamento em tempo real, não levando mais de 2 segundos para geração do arquivo de saída (RNF);
g) ser implementado na linguagem de programação *Object Pascal* (RNF);
h) ser desenvolvido no ambiente de programação Delphi (RNF);
i) funcionar no sistema operacional Windows XP ou superior (RNF).

3.2 ESPECIFICAÇÃO

Nesta seção são apresentadas a relação dos fonemas suportados pelo protótipo e a especificação do formato do arquivo de entrada. Também são descritos os diagramas de casos de uso, de classes e de sequência da *Unified Modeling Language* (UML) especificados utilizando a ferramenta *Enterprise Architect*. Por fim, é explicado o formato do arquivo de saída (formatação da transcrição).

3.2.1 Lista de fonemas

Em um sistema texto-fala a especificação dos fonemas suportados implica diretamente no resultado final do programa. Os fonemas utilizados no protótipo desenvolvido são baseados em um banco de difones[5] encontrado e disponibilizado pelo sintetizador MBROLA[6] (DUTOIT et al., 2005), descrito posteriormente na seção que trata das técnicas e ferramentas utilizadas.

A partir do banco de fonemas denominado BR3 (COSTA, 2005), para cada fonema encontrado no banco, foi feito um mapeamento de regras gramaticais e também de regras baseadas nas limitações do sintetizador. Assim, no quadro 2 é apresentada a lista de fonemas suportados pelo protótipo desenvolvido, sendo que para cada fonema tem-se: um exemplo de palavra que o contenha, a transcrição correspondente, o tempo ideal para síntese do fonema e uma breve descrição da regra definida para identificá-lo. A partir deste quadro já é possível verificar alguns aspectos do formato de saída.

---

[5] Os difones "são pequenas seqüências [sic] de áudio que mostram a transcrição da metade de um fonema para a metade do outro" (MACHADO, 2007), desta forma concatenando-os linearmente.

[6] Este sintetizador é um projeto iniciado em 1995 pelo *TCTS Lab* da *Faculté Polytechnique de Mons* (Bélgica). Foi incorporado ao protótipo desenvolvido por disponibilizar gratuitamente um banco de fonemas no português brasileiro (BR3) e por ser um software reconhecido internacionalmente pela comunidade científica.

fonema	exemplo	transcrição	tempo	identificar na ocorrência de
/b/	***b**arco*	"ba r2 ku	105	b
/k/	***c**om*	"k om	105	ca, co, cu
/d/	***d**oce*	"dose	100	d
/g/	***g**rande*	"gr am de	80	ga, gue, gui, go, gu, gua, g+consoante
/p/	***p**ai*	"pai	110	p
/t/	***t**aco*	"tako	100	t
/f/	***f**ácil*	"fasiw	60	f
/v/	***v**inho*	"vi nh o	60	v
/j/	***j**ato*	"jato	60	j, ge, gi
/s/	***s**ala*	"sala	90	s, ss, sc, sç, x+consante, ce, ci
/s2/	*ca**s**ca*	ka "s2 ka	90	s+consante
/x/	***ch**ave*	"chave	100	x, ch
/z/	*a**s**a*	"aza	95	z, s entre duas vogais
/m/	***m**esmo*	"me s2 mo	80	m
/n/	***n**unca*	"n um ka	80	n
/nh/	*gali**nh**a*	ga li "nh a	90	nh
/l/	***l**anche*	"l am xe	70	l
/h/	*a**lh**o*	"a lh o	90	lh
/r/	*pu**r**o*	"puro	60	r
/r2/	*a**r**pa*	"a r2 pa	60	r$
/rr/	*to**rr**e*	"to rr e	70	rr
/a/	*v**a**le*	"vale	100	a
/@/	*tam**a**nho*	ta "m@ nh o	110	an
/am/	*c**am**panha*	k am "pa nh a	100	am, ão
/e/	*p**ê**ra*	"pera	90	e, ea, ê
/ee/	*qu**e**ro*	"k ee ro	90	é
/em/	*qu**en**te*	"k em te	90	en, em, õe
/i/	*p**i**co*	"piko	80	i
/im/	*br**in**co*	"br im ko	70	in, im
/o/	*t**o**lo*	"tolo	90	o
/oo/	*b**o**la*	"b oo la	90	ó
/om/	***om**bro*	"om bro	90	om, on
/u/	*d**u**ro*	"duro	40	u
/um/	*alg**um***	aw"g um	110	um
/y/	*m**ai**s*	"may s2	80	i+vogal, vogal+i
/w/	*m**au***	"maw	95	u+vogal, vogal+u

Quadro 2 – Fonemas suportados pelo software, extraídos do banco de fonemas BR3

Alguns dos fonemas suportados não apresentam nenhuma regra especial para serem transcritos, como no caso dos fonemas correspondentes às letras ***b***, ***d***, ***p***, ***t***, ***f*** e ***v***, em que a letra, singularmente, já representa o fonema. Os encontros consonantais ***nh*** e ***lh*** também não necessitam de nenhuma regra para transcrição. Contudo, para os demais fonemas torna-se necessário o uso de algumas regras para uma correta transcrição. Algumas regras estão diretamente ligadas à gramática da língua portuguesa e outras estão ligadas às limitações do sintetizador MBROLA. As regras de transcrição para os fonemas suportados encontram-se no

Apêndice A.

3.2.2 Formato de entrada

Para classificar sistematicamente os elementos que possuem características fonéticas em um texto escrito em português, foi necessário definir uma gramática[7] compatível com este intento. Desta forma, as definições feitas para o formato de entrada foram norteadas pela qualidade fonética dos signos e sua disposição regular nos textos do idioma nacional.

Com isso, para que o texto de entrada seja corretamente transcrito pelo protótipo, o mesmo deve seguir o formato especificado nesta gramática. O quadro 3 apresenta as definições regulares que restringe o alfabeto suportado pelo protótipo para a formação das palavras.

```
maiuscula: [A-Z]
minuscula: [a-z]
acentuada_min: [àáéíóúâêôãõüç] //Notações léxicas - Pg. 31 CEGALLA
acentuada_mai: [ÀÁÉÍÓÚÂÊÔÃÕÜÇ] //Notações léxicas - Pg. 31 CEGALLA
palavra: ({maiuscula} | {acentuada_mai})?
         ({minuscula} | {acentuada_min})+ | ({maiuscula} | {acentuada_mai})
algarismo: [0123456789]
num_inteiro: {algarismo}+
```

Quadro 3 – Definições regulares da gramática

Observa-se que são suportados os seguintes caracteres:

a) todas as letras do alfabeto, maiúsculas e minúsculas, acentuadas ou não, incluindo a cedilha;
b) todos os algarismos;
c) alguns sinais de pontuação, tais como vírgula, ponto e vírgula, exclamação, interrogação, dois pontos, travessão, reticências, ponto final e parênteses.

No quadro 4 é apresentada a definição para formação das palavras (sequências de caracteres suportados) que constitui uma classificação própria para o protótipo. Também é nesta parte que estão especificados os demais caracteres que devem ser ignorados no texto de entrada.

[7] A gramática neste caso não se refere à gramática da língua portuguesa, mas à especificação do padrão de formação de palavras, dos caracteres de pontuação reconhecidos e dos caracteres que devem ser ignorados, bem como da sequência em que as palavras e caracteres de pontuação podem ser combinados para formar frases.

```
//Sequência de caracteres suportados (tokens)
palavra_simples: {palavra}
palavra_composta: {palavra}("-"{palavra})+
sigla: ({acentuada_mai} | {maiuscula}) ({acentuada_mai} | {maiuscula})+
numero: {num_inteiro}("."{num_inteiro})* | {num_inteiro}(","{num_inteiro})*
pontuacao: "," | ";" | "!" | "?" | ":" | "-" | "..."

"."
"("
")"

//Abreviaturas de unidades federativas
AC = sigla: "AC" //Acre
AM = sigla: "AM" //Amazonas
AL = sigla: "AL" //Alagoas
AP = sigla: "AP" //Amapá
BA = sigla: "BA" //Baía
CE = sigla: "CE" //Ceará
DF = sigla: "DF" //Distrito federal
ES = sigla: "ES" //Espírito Santo
GO = sigla: "GO" //Goiás
MG = sigla: "MG" //Minas Gerais
MT = sigla: "MT" //Mato Grosso
PA = sigla: "PA" //Pará
PB = sigla: "PB" //Paraíba
PE = sigla: "PE" //Pernambuco
RJ = sigla: "RJ" //Rio de Janeiro
RN = sigla: "RN" //Rio grande do Norte
RS = sigla: "RS" //Rio grande do Sul
SC = sigla: "SC" //Santa Catarina
SE = sigla: "SE" //Sergipe
SP = sigla: "SP" //São Paulo

//Siglas especiais
TV = sigla: "TV" //Televisão
PM = sigla: "PM" //Polícia Militar
DVD = sigla: "DVD" // Digital Video Disc
CD = sigla: "CD" //Compact disc
RW = sigla: "RW" //regravável
PC = sigla: "PC" //Personal computer

//Números romanos
I = palavra_simples: "I" //Numero romano 1
II = sigla: "II" //Numero romano 2
III = sigla: "III" //Numero romano 3
IV = sigla: "IV" //Numero romano 4
V = palavra_simples: "V" //Numero romano 5
VI = sigla: "VI" //Numero romano 6
VII = sigla: "VII" //Numero romano 7
VIII = sigla: "VIII" //Numero romano 8
IX = sigla: "IX" //Numero romano 9
X = palavra_simples: "X" //Numero romano 10

//Símbolos ignorados
:[\s \r \n \t \\ / \+ \* \[ \] \" \34 \35 \36 \37 \38 \39 \123 \124 \125 \64 "ª"
"º" \248 "´" "`" "’" ]
```

Quadro 4 – Definição das sequências de caracteres

A classificação especificada pode ser resumida da seguinte maneira:

a) palavra simples: palavra formada apenas por letras;

b) palavra composta: palavra formada por letras em que se emprega o hífen;

c) sigla: palavra formada somente com letras maiúsculas;

d) sigla de unidade federativa do Brasil: siglas dos estados brasileiros;

e) siglas especiais: siglas frequentemente utilizadas;

f) número: sequência de algarismos, intercalados ou não por apenas uma vírgula ou um ponto;

g) número romano: números romanos de um até dez.

h) pontuação: sinais de pontuação relacionados anteriormente;

i) parênteses (abertura e fechamento).

No quadro 5 são especificadas as regras sintáticas do texto de entrada. Pode-se observar na linha 04, uma regra destinada a frases escritas entre parênteses. Esta especificação sintática serve para qualificar as locuções encontradas entre estes símbolos, que segundo Cegalla (1985, p. 86), quando usadas “para isolar palavras, [...] ou frases intercaladas no período, com caráter explicativo, [...] são proferidas em tom mais baixo”. Para isso, é importante que o texto de entrada respeite o agrupamento destes símbolos.

```
01  <frase>::=  <romano> <sinal> <frase_>
02            | <palavra> <sinal> <frase_>
03            | <sigla> <sinal> <frase_>
04            | "(" <frase> ")" <sinal> <frase_> //parenteses CEGALLA pg. 87
05            | <numeral> <sinal> <frase_> ;
06
07  <frase_>::= ε| <frase> ;
08  <numeral> ::= numero ;
09  <palavra>::= palavra_simples | palavra_composta ;
10  <sinal>::= ε | pontuacao | ".";
11  <sigla> ::= sigla
              | AC | AM | AL | AP | BA | CE | DF | ES | GO | MG | MT | PA | PB
              | PE | RJ | RN | RS | SC | SE | SP | TV | PM | DVD | RW | CD | PC ;
12  <romano> ::= I | II | III | IV | V | VI | VII | VIII | IX | X;
```

Quadro 5 – Definição sintática da gramática

3.2.3 Diagrama de casos de uso

O protótipo desenvolvido possui quatro casos de uso: `Informa texto de entrada`, `Efetua processamento e transcrição`, `Informa separação silábica errada` e `Informa transcrição inadequada`, como pode ser observado na figura 6. Há uma sequência obrigatória para os casos de uso: primeiro deve-se informar o texto de entrada e após realizar o processamento linguístico das palavras e a transcrição fonética. Ainda, pode-se informar uma separação silábica incorreta ou uma transcrição inadequada de uma palavra.

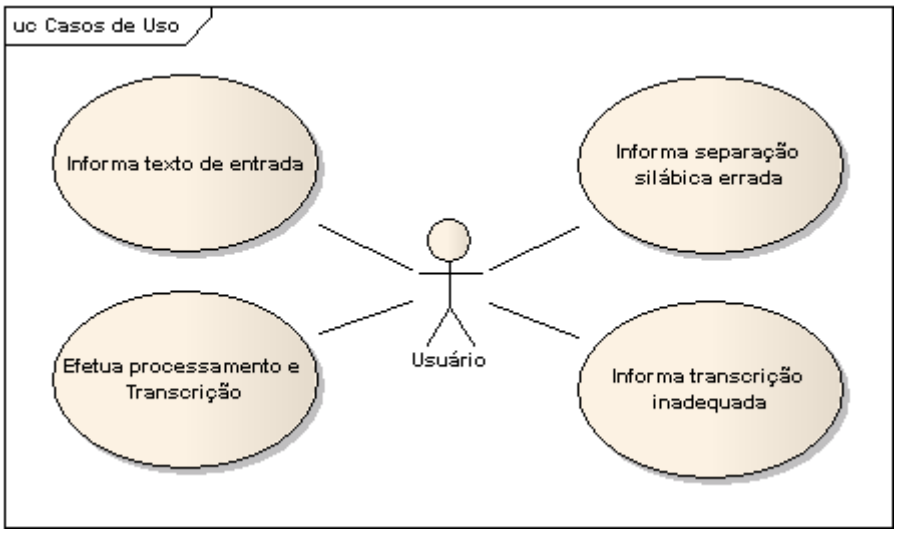

Figura 6 - Diagrama de casos de uso

O caso de uso `Informa texto de entrada`, detalhado no quadro 7, é a ação inicial do usuário no protótipo. O texto informado deve obrigatoriamente ser escrito em português. Caso seja inserido um texto em outra língua, é possível que ocorram problemas, podendo acarretar na geração de um arquivo de áudio incompleto e, consequentemente, uma síntese do texto de entrada inapropriada. Contudo, alguns termos aportuguesados que fazem parte do hábito linguístico brasileiro, podem ser escritos, como por exemplo, as palavras *software*, *show*, *web* ou *site*.

UC01 – Informa texto de entrada.	
Cenário principal	01) O usuário informa o texto no primeiro campo da tela.
Pós-condições	Um texto de entrada deve ter sido digitado.

Quadro 7 – Detalhamento do caso de uso `Informa texto de entrada`

O caso de uso `Efetua processamento e transcrição` (quadro 8) representa as ações de processamento do texto informado e a transformação das palavras em um código intermediário que represente finalmente a transcrição fonética do texto nos formatos do sintetizador suportado. Também pode ser acionado o sintetizador MBROLA para realizar o processamento acústico. Este caso de uso apresenta duas exceções: caso não sejam respeitadas as limitações de textos compatíveis com o protótipo, ou ainda se ocorrer uma separação silábica incorreta de alguma palavra, gerando sequências de fonemas inválidos.

UC02 – Efetua processamento e transcrição	
Pré-condições	Ter um texto de entrada digitado.
Cenário principal	01) O usuário clica no botão `Processar palavras`. 02) O protótipo apresenta no segundo campo da tela uma lista das palavras encontradas, classificadas de acordo seu tipo e acompanhadas, quando aplicável, da separação silábica e do número de letras em cada sílaba. 03) O usuário analisa as separações silábicas e clica no botão `Transcrever`. 04) O protótipo apresenta no terceiro campo da tela, a transcrição fonética do texto (lista de fonemas), acompanhada do tempo de duração e de informação prosódica de entonação de cada fonema. 05) O usuário clica no botão `Sintetizar`. 06) O protótipo lê a lista gerada no passo 04, gera o arquivo `Transcrição.pho` no mesmo local do executável, gera um arquivo *Wave* com a verbalização do texto e executa o arquivo *Wave*.
Exceção 01	No passo 02, caso o texto informado viole uma regra sintática da gramática definida (como por exemplo, abrir parênteses e não fechar), uma mensagem de erro sintático será emitida.
Exceção 02	No passo 06, caso o texto possua algum problema na separação silábica ou tenha sido transcrito com problema (uma sequência impronunciável de fonemas), uma mensagem de erro indicando o par de fonemas inválido será emitida.
Pós-condições	O arquivo de transcrição e o arquivo de som devem ter sido criados.

Quadro 8 – Detalhamento do caso de uso `Efetua processamento e transcrição`

Caso seja identificado que o protótipo tenha feito uma separação silábica incorreta, o usuário pode alimentar o dicionário silábico, informando a separação correta da palavra. O dicionário silábico é então atualizado e utilizado no próximo processamento de palavras. O detalhamento desse caso de uso é apresentado no quadro 9.

UC03 – Informa separação silábica errada.	
Pré-condições	Ter um texto de entrada processado e transcrito.
Cenário principal	01) O usuário seleciona no primeiro campo da tela uma palavra que teve a separação silábica incorreta. 02) O usuário clica no botão `Separação silábica errada`. 03) O protótipo apresenta uma tela com um nova caixa de texto com a palavra selecionada. 04) O usuário digita as sílabas separadas por hífen conforme descreve a gramática da língua portuguesa. 05) O usuário clica no botão `Ok` para confirmar a nova separação silábica ou no botão `Cancel` para sair sem gravar a nova separação para a palavra.
Pós-condições	A nova separação silábica para a palavra deve ser gravada no dicionário silábico e será utilizada quando for realizado novo processamento das palavras.

Quadro 9 – Detalhamento do caso de uso `Informa separação silábica errada`

Da mesma forma que pode ser informada a separação silábica correta, a transcrição de uma palavra também pode ser corrigida. Assim, o usuário pode alimentar o dicionário de exceções, que guarda as palavras transcritas inadequadamente. O dicionário é atualizado e o termo transcrito corretamente na próxima transcrição executada. O detalhamento do caso de uso `Informa transcrição inadequada` é apresentado no quadro 10.

UC04 – Informa transcrição inadequada.	
Pré-condições	Ter um texto de entrada processado e transcrito.
Cenário principal	01) O usuário seleciona no primeiro campo da tela uma palavra que teve a sua transcrição inadequada. 02) O usuário clica no botão `Transcrição inadequada`. 03) O protótipo apresenta uma tela com um nova caixa de texto contento a transcrição atual da palavra. 04) O usuário modifica a transcrição, respeitando os símbolos de fonemas suportados. 05) O usuário clica no botão `Gravar no dicionário` ou fecha a janela para sair sem gravar a nova transcrição para a palavra.
Pós-condições	A nova transcrição para a palavra deve ser gravada no dicionário de exceções e será utilizada quando for realizado novo processamento das palavras.

Quadro 10 – Detalhamento do caso de uso `Informa transcrição inadequada`

3.2.4 Diagrama de classes

O diagrama de classes fornece uma visão de como está estruturado o protótipo e apresenta como as classes estão relacionadas entre si. As classes responsáveis pelo processamento e transcrição textual foram agrupadas em um pacote, denominado FurbTTS. Este pacote gera um arquivo com extensão BPL, que é implementado pelo executável do protótipo. O diagrama da figura 11 apresenta somente as classes que se encontram dentro deste pacote. As demais classes, que fazem a interação com o usuário, não são descritas.

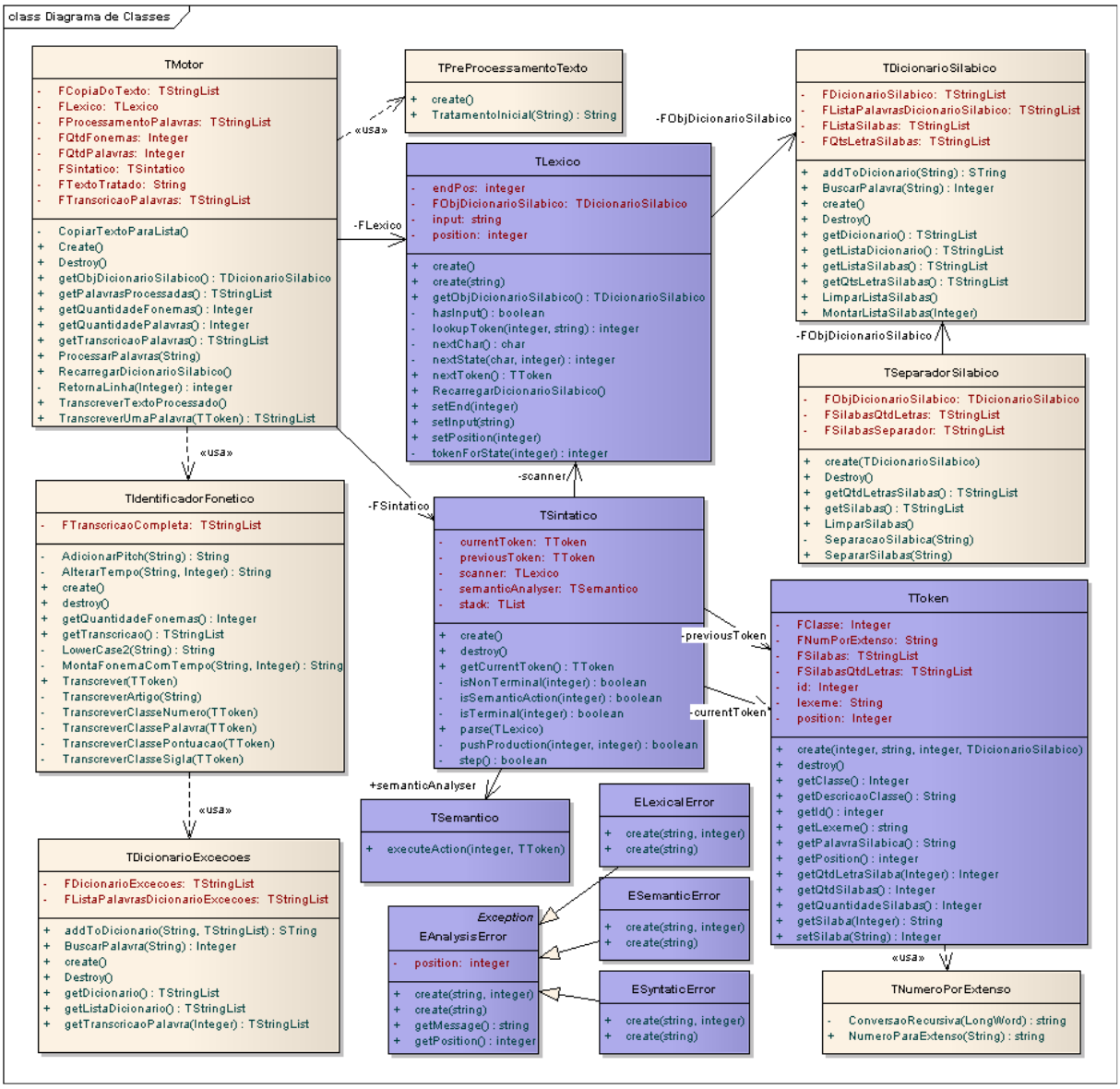

Figura 11 - Diagrama de classes

A classe principal é denominada `TMotor`, em alusão a palavra *engine* utilizada para definir o núcleo reaproveitável de alguns jogos de computador . Esta classe é responsável por acionar todas as demais classes e é a única que deve ser implementada pelos projetos de interface de interação com usuário. Nesta classe mantém-se em memória: o texto tratado pela classe `TPreProcessamentoTexto`, a lista completa do processamento das palavras e a lista completa da transcrição fonética das palavras, que é obtida da classe `TIdentificadorFonetico`.

A classe `TPreProcessamentoTexto` é a segunda classe para o processamento das palavras. Nesta classe é realizado o tratamento inicial do texto, que consiste em: transformar abreviaturas em sua forma por extenso, identificar caracteres que serão ignorados pelo analisador léxico e substituí-los por palavras quando necessário, e eliminar conjuntos de

caracteres que possam gerar problemas sintáticos.

As classes `TLexico`, `TToken`, `TSintatico`, `TSemantico`, `EAnalysisError`, `ELexicalError`, `ESyntaticError` e `ESemanticError` foram geradas pela ferramenta GALS e alteradas de acordo com a especificidade do protótipo proposto. As classes `TLexico` e `TSintatico` fazem, respectivamente, as análises léxica e sintática do texto tratado pelo pré-processamento. O analisador léxico instancia objetos `TToken` para cada palavra reconhecida no texto, realiza a classificação da palavra e efetua a separação silábica através da classe `TSeparadorSilabico`. Ainda no momento da criação do objeto `TToken`, caso a palavra seja classificada como um número, a classe `TNumeroPorExtenso` faz a tradução deste número para sua descrição por extenso.

As classes `TDicionarioExcecoes` e `TDicionarioSilabico` são usadas para controlar os arquivos que mantêm estes dicionários, implementando os métodos que procuram por palavras, adicionam novas palavras nos arquivos e mantêm os arquivos em memória durante a execução do programa.

3.2.5 Diagramas de sequência

Esta seção apresenta os diagramas de sequência que representam o conjunto de passos que o protótipo executa para realizar determinada tarefa, com base nas ações do usuário. Os diagramas representam o caso de uso `Efetua processamento e transcrição`, que é o caso mais relevante. Por se tratar de um caso de uso extenso, foi dividido em dois diagramas, que representam, respectivamente: o processamento linguístico das palavras e a transcrição das palavras. No diagrama da figura 12 é apresentado o processamento linguístico, onde é realizado um pré-processamento do texto de entrada, que é enviado para o analisador léxico e, posteriormente, para o analisador sintático. Por fim, são criados objetos `TToken` para cada palavra reconhecida com suas respectivas classificação e separação silábica.

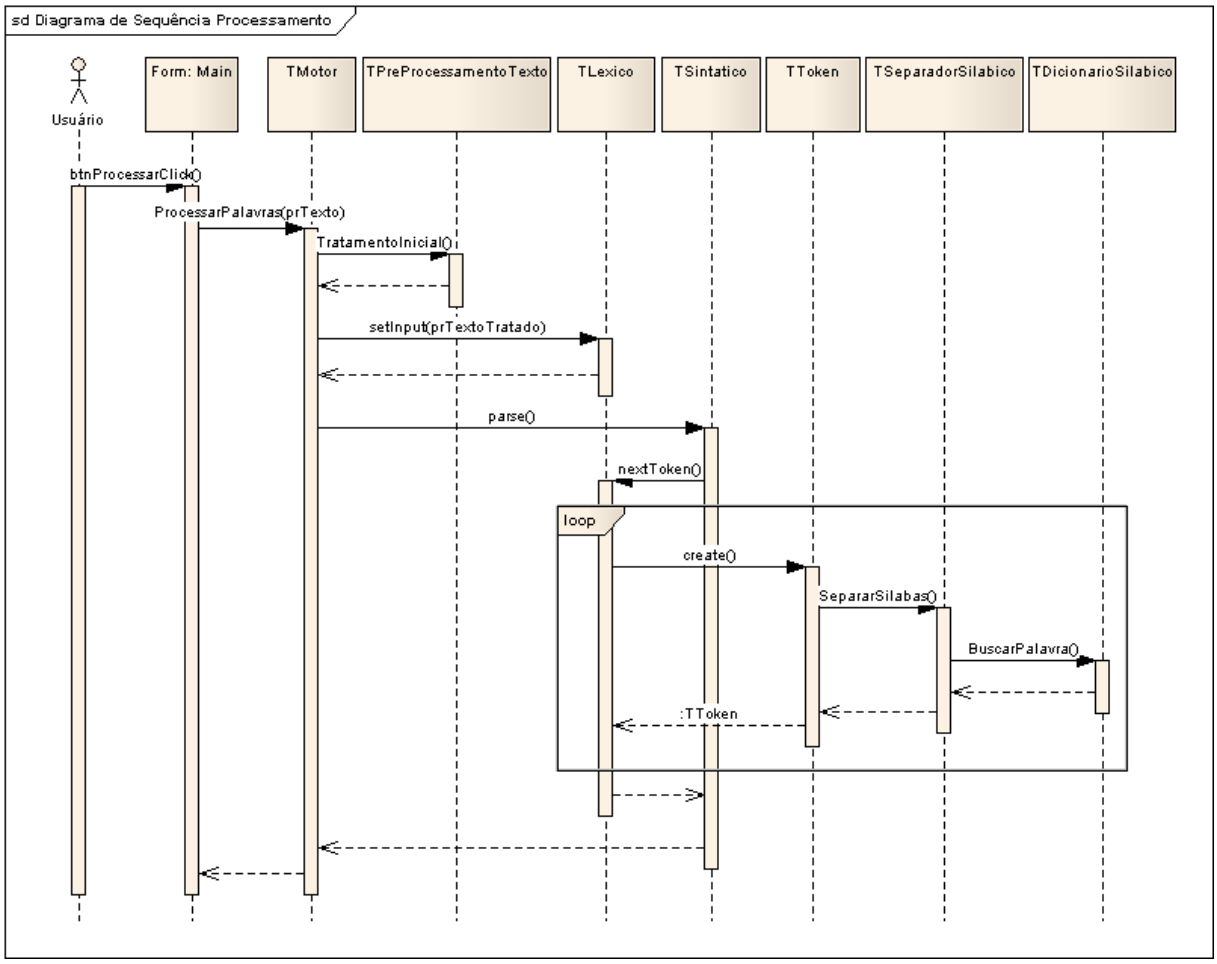

Figura 12 – Diagrama de sequência do processamento linguístico das palavras

A figura 13 apresenta os passos para a transcrição fonética das palavras processadas. Nesta etapa observa-se que os objetos `TToken` são recriados pelo analisador léxico. Isso deve-se ao fato de a classe `TLexico` criar os objetos em tempo de execução, ao invés de mantê-los em memória. Esta característica é proveniente da arquitetura gerada pelo GALS.

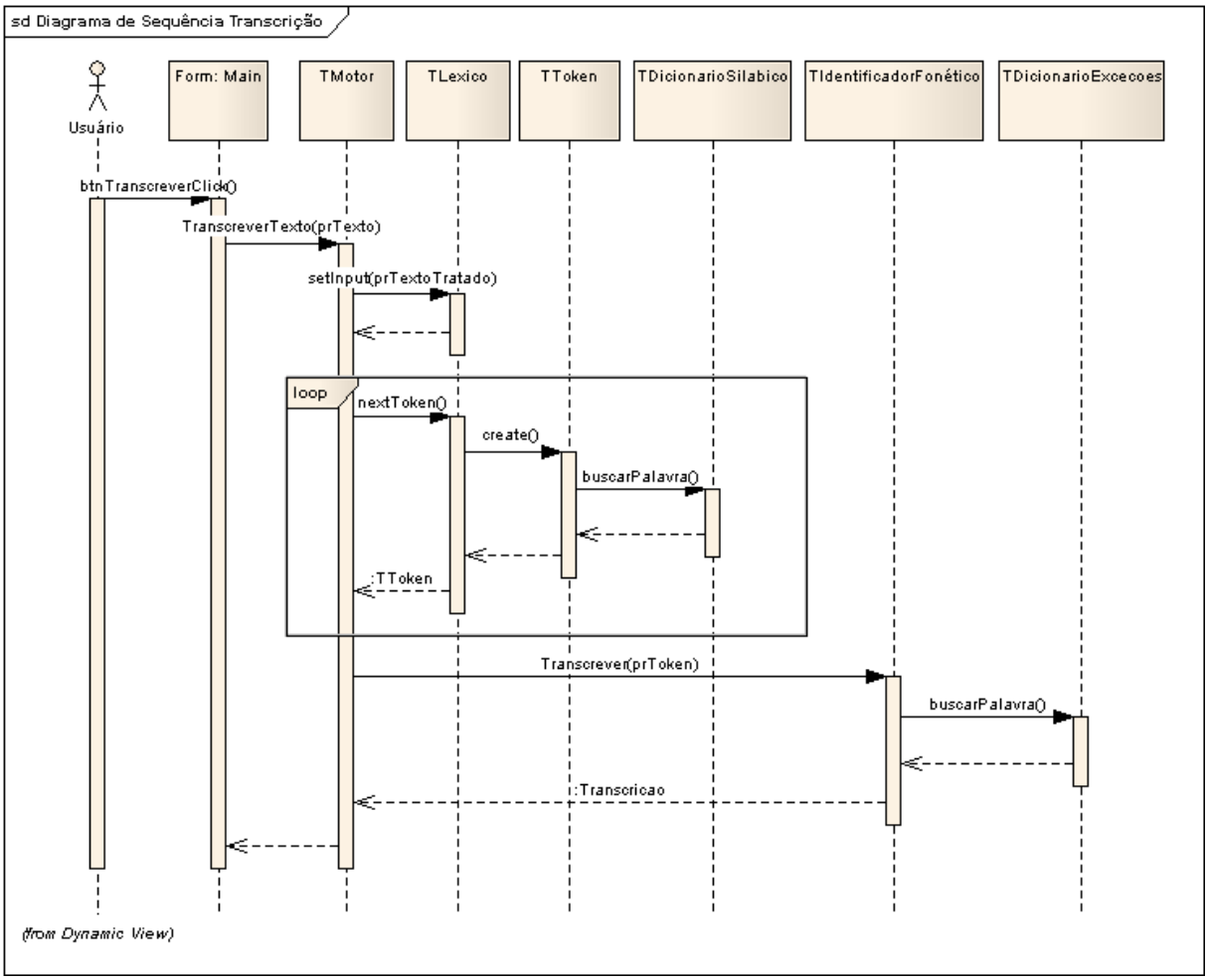

Figura 13 – Diagrama de sequência da transcrição das palavras

3.2.6 Formato de saída

Um dos objetivos desse trabalho é gerar um arquivo de saída com a lista de fonemas que devem ser sintetizados. O formato de saída foi especificado obedecendo ao formato de entrada do sintetizador MBROLA, onde cada fonema deve ser acompanhado de um tempo de duração, separado por espaço em branco e, opcionalmente, de alguma informação prosódica, também separada por espaço em branco.

O tempo de duração do fonema corresponde ao tempo em milissegundos que o fonema é pronunciado pelo sintetizador. Não há limite definido para o tempo de duração de um fonema, porém, devido aos testes realizados, recomenda-se que para uma fala natural, este tempo não exceda 150 milissegundos.

A informação prosódica diz respeito à alteração no *pitch* ou frequência fundamental do som. Esta informação é importante para sinalizar ao sintetizador mudanças de entonação, que

ocorrem principalmente com a pontuação do texto. Quando uma frase é exclamada, por exemplo, aumenta-se a entonação ao final da mesma. A informação prosódica permite realizar este mesmo efeito, sendo composta por dois números inteiros que indicam, respectivamente, um percentual dentro do tempo definido do fonema e o valor da frequência em hertz (Hz). Esta alteração de frequência pode ser repetida até vinte vezes para o mesmo fonema. Contudo, para uma fala natural, dois conjuntos de informações prosódicas já se mostram suficientes. O sintetizador permite até vinte, pois está preparado para simular canções.

O quadro 14 mostra um exemplo de transcrição para a frase “Ataque!”, com informação prosódica nos fonemas /a/, /t/ e /e/. No caso do fonema /a/, a saída apresentada no quadro 14 indica ao sintetizador para adicionar um ponto de mudança de frequência de 86 Hz, aos 10% de 100 milissegundos, e outra mudança na freqüência de 82 Hz aos 90% dos mesmos 100 milissegundos. Desta forma, a frequência fará uma curva de alteração em cada uma das informações prosódicas.

```
a 100 10 86 90 82
t 100 50 90
a 100 10 86 90 82
k 105
e 90 10 110
_ 5 10 90
_ 400 90 110
```

Quadro 14 – Transcrição da frase “Ataque!”

3.3 IMPLEMENTAÇÃO

Nesta seção são apresentadas informações sobre as técnicas e ferramentas utilizadas para a implementação, bem como a implementação propriamente dita de cada etapa do protótipo. Também é nesta seção que é apresentado o sintetizador escolhido para este trabalho. Por fim, é descrita a operacionalidade do protótipo desenvolvido.

3.3.1 Técnicas e ferramentas utilizadas

Foram utilizadas algumas ferramentas de desenvolvimento de software e aplicadas algumas técnicas e algoritmos.

Primeiramente, foi utilizada a ferramenta GALS para a especificação da gramática

suportada, apresentada anteriormente na seção 3.2.2. Apesar de ter sido criado para gerar analisadores léxicos e sintáticos de linguagens de programação, o GALS mostrou-se adequado para gerar código capaz de analisar a estrutura de um texto e separá-lo em palavras, caracteres de pontuação ou números.

A partir da especificação da gramática, foi gerado automaticamente código que implementa as análises léxica e sintática. O código foi gerado segundo o paradigma da orientação a objetos e por isso toda a estrutura de classes foi mantida, necessitando apenas a adição de alguns métodos e propriedades. As classes geradas pelo GALS foram adicionadas a uma biblioteca, que é utilizada no protótipo.

Dentro desta biblioteca encontra-se a classe `TSeparacaoSilabica`, que é responsável por criar a lista das sílabas identificadas em cada palavra, classificada como palavra simples ou palavra composta. Esta classe implementa um algoritmo encontrado e adaptado de Gomes (1998). O funcionamento do algoritmo (Apêndice B) baseia-se na análise da próxima letra a partir da primeira letra da palavra. Ao identificar o final de uma sílaba, ela é adicionada a uma lista e mantida em memória para ser usada mais tarde pelo identificador fonético.

Existe também um algoritmo (Apêndice C) implementado nesta biblioteca, responsável pela tradução de um número para sua forma por extenso (biunívoca cardinal). Fundamentalmente, o algoritmo consiste em realizar a divisão do número por uma faixa de valor e o resto desta divisão resulta em uma nova faixa. Este processo é então executado recursivamente, até que o resto da divisão seja menor que dezenove. Este algoritmo está implementado na classe `TNumeroPorExtenso` e foi encontrado e adaptado de *site* DevMedia Group (2009).

No desenvolvimento do protótipo também foi utilizado o sintetizador MBROLA, cujo projeto de mesmo nome pretende colecionar um conjunto de sintetizadores de fala para o máximo de idiomas possível e disponibilizá-los para uso gratuito em aplicações não comerciais (DUTOIT et al., 2005). O sintetizador está baseado na concatenação de difones. Utiliza como entrada uma lista de fonemas (em forma textual), juntamente com alguma informação prosódica (duração do fonema, descrição linear de entonação), e produz amostras em 16 bits (linear), na frequência do banco de difones utilizado. Portanto, este projeto não se trata de um sistema *Text-To-Speech* (texto-fala), visto que ele não aceita texto bruto como entrada.

Os bancos de difones são trabalhados para o formato do sintetizador e são necessários para executar o programa. Estes bancos são disponibilizados no *site* do projeto e estão organizados de forma a fomentar outras organizações a compartilhar seus bancos de difones.

A instalação do MBROLA requer apenas o uso de uma biblioteca (`mbrplay.dll`) que deve acompanhar o sistema que se utilizar do sintetizador .

3.3.2 Pré-processamento do texto

A etapa de pré-processamento do texto tem por objetivo tratar o texto de entrada para evitar que erros léxicos e sintáticos ocorram. Para isso, o pré-processamento consiste fundamentalmente em identificar abreviaturas e expandi-las para sua forma por extenso; identificar caracteres que serão ignorados pelo analisador léxico e substituí-los por palavras quando necessário; e eliminar conjuntos de caracteres que podem causar problemas.

As abreviaturas que o protótipo consegue identificar são limitadas e foram obtidas arbitrariamente de uma gramática da língua portuguesa (BARBOSA, 1973, p. 33). As abreviaturas devem ser obrigatoriamente seguidas de ponto final, exceto quando determinam símbolos científicos, como por exemplo, o termo km ou kg. O quadro 15 traz a lista das abreviaturas suportadas.

abreviatura	forma extensa	abreviatura	forma extensa	abreviatura	forma extensa
apto	Apartamento	dt	Data	lat	Latitude
apart	Apartamento	div	Divisão	ltda	Limitada
alm	Almirante	doc	Documento	pgto	Pagamento
arr	Arroba	dz	Dúzia	proc	Processo
av	Avenida	ed	Edição	prof	Professor
bel	Bacharel	ex	Exemplo	pres	Presidente
cap	Capital	fat	Fatura	rel	Relatório
cfm	Conforme	fig	Figura	remte	Remetente
cg	Centigrama	fut	Futuro	sr	Senhor
cia	Companhia	gen	General	sra	Senhora
cl	Classe	hab	Habitante	srta	Senhorita
cm	Centímetro	inf	Informação	tel	Telefone
cmt	Comandante	ind	Índice	tit	Título
cop	Copiado	km	Quilômetro	vcto	Vencimento
cv	Cavalo	kg	Quilograma		

Quadro 15 – Abreviaturas suportadas

Caracteres que podem ter um significado importante para o sentido da frase e que serão ignorados pelo analisador léxico também são tratados no pré-processamento. Como exemplo disso pode-se citar a sequência R$ que indica a palavra *reais*. O cifrão é um caractere que não é processado pelo protótipo e por isso esta sequência de caracteres seria identificada apenas como *R*. Outro caractere ignorado é o símbolo % que deve ser pronunciado *porcento*.

Por fim, existem conjuntos de caracteres que também podem causar erros no analisador sintático, por exemplo um ponto seguido de travessão. Neste exemplo, o travessão, que não tem função fonética, é eliminado do texto. No quadro 16 estão exemplos de tratamentos realizado na etapa de pré-processamento.

```
//-- Tratamento de abreviaturas
if pos(' dt.',LowerCase(Result)) > 0 then begin
  Result := StringReplace(Result,' dt.','data',[rfReplaceAll, rfIgnoreCase]);
end;

if pos(' cm ',LowerCase(Result)) > 0 then begin
  Result := StringReplace(Result,' cm ',' centímetros ',[rfReplaceAll,
rfIgnoreCase]);
end;

if pos(' ed.',LowerCase(Result)) > 0 then begin
  Result := StringReplace(Result,' ed.','edição',[rfReplaceAll, rfIgnoreCase]);
end;

//-- Símbolos ignorados pelo analisador léxico, mas com importância fonética
if pos('r$',LowerCase(Result)) > 0 then begin
  Result := StringReplace(Result,'R$','Reais',[rfReplaceAll, rfIgnoreCase]);
end;

if pos('%',LowerCase(Result)) > 0 then begin
  Result := StringReplace(Result,'%',' porcento ',[rfReplaceAll, rfIgnoreCase]);
end;

//-- Conjuntos de símbolos que causam erro sintático
if pos(',-',LowerCase(Result)) > 0 then begin
  Result := StringReplace(Result,',-',',',[rfReplaceAll, rfIgnoreCase]);
end;

if pos('. -',LowerCase(Result)) > 0 then begin
  Result := StringReplace(Result,'. -','.',[rfReplaceAll, rfIgnoreCase]);
end;

if pos(') -',LowerCase(Result)) > 0 then begin
  Result := StringReplace(Result,') -',')',[rfReplaceAll, rfIgnoreCase]);
end;

//-- Encontro consonantal que pode ser encontrado em grafias antigas
//-- da língua portuguesa
if pos('ph',LowerCase(Result)) > 0 then begin
  Result := StringReplace(Result,'ph','f',[rfReplaceAll, rfIgnoreCase]);
end;
```

Quadro 16 – Exemplos de tratamento de texto

3.3.3 Analisador linguístico

O analisador linguístico do programa está implementado junto ao analisador léxico. A classificação das palavras não ocorre de acordo com a gramática da língua portuguesa. Ao invés disso, foi definida uma classificação baseada nas regras da gramática formal especificada no GALS (seção 3.2.2). Esta classificação, no entanto, não tem o propósito de orientar a pronúncia da palavra, mas sim auxiliar a próxima etapa, passando toda a responsabilidade da pronúncia para o identificador fonético.

A estratégia implementada, portanto, não permite que uma palavra possua duas pronúncias diferentes, como por exemplo a palavra *piloto* nas sentenças “O *piloto* saltou.” e

“Eu *piloto* carros.”. Para que este problema pudesse ser corrigido, seria necessário um banco de palavras da língua portuguesa com suas classificações gramaticais, uma vez que a palavra *piloto* é substantivo na primeira frase e verbo na segunda. Para um usuário que consegue identificar esta diferença gramatical, já é o suficiente para mudar sua pronúncia. Porém, para que o protótipo pudesse pronunciar a palavra corretamente nos dois casos, seria necessário armazenar a sua pronúncia juntamente com a classificação gramatical. Isto se mostrou impraticável, uma vez que este banco de palavras teria que ser criado e populado para que o protótipo pudesse funcionar adequadamente. Desta forma, pode-se afirmar que o protótipo desenvolvido só permite uma única pronúncia para cada palavra.

Ainda sob a responsabilidade do analisador linguístico está a separação silábica das palavras. A separação silábica ocorre dentro do analisador léxico sempre que é instanciado um objeto `TToken` cuja classificação seja dada como palavra simples, palavra composta ou número. No caso de números, antes da separação silábica, ocorre a tradução do mesmo para sua forma biunívoca cardinal. Portanto, os números nunca serão transcritos para sua forma ordinal. “Os números ordinais, ou simplesmente ordinais, são números usados para assinalar uma posição numa sequência ordenada: primeiro, segundo, terceiro, quarto, quinto, sexto etc.” (MEIER, 2009). Para esta tradução o analisador linguístico implementa um algoritmo, cujo valor numérico limite é 4.294.967.295 (quatro bilhões, duzentos e noventa e quatro milhões, novecentos e sessenta e sete mil, duzentos e noventa e cinco). Caso o número possua uma parte fracionária, a mesma é transcrita em sua forma cardinal, separando-se da parte inteira pela palavra *vírgula*.

A separação silábica é de extrema importância, pois orienta a próxima etapa do protótipo na escolha dos fonemas corretos. O algoritmo implementado foi melhorado em vários aspectos e apresenta um desempenho muito satisfatório com poucos problemas. Para exemplificar a importância deste processo, tem-se o caso dos fonemas /m/ e /n/, que só podem ser transcritos quando se encontram em início de sílaba. “Caso contrário, as letras m e n são simples sinais de nasalização da vogal ou do ditongo anterior” (CEGALLA, 1997, p. 29). Outro exemplo da importância da separação silábica pode ser observado com a palavra *suíno*, que o protótipo não consegue fazer a separação silábica. Isso se deve à forma como o algoritmo está implementado, que apesar de ter uma margem de erro muito pequena, não consegue identificar as sílabas desta palavra. Neste exemplo, caso a separação silábica não seja corrigida, a palavra será transcrita conforme o quadro 17. A letra ***í*** seria transformada na sua nasal, destruindo completamente o sentido da palavra.

```
s 90
u 40
y 80
o 90
```

Quadro 17 – Transcrição inadequada da palavra *suíno*

Para solucionar este caso, a palavra *suína* deve ser adicionada ao dicionário silábico, que armazena a separação silábica correta das palavras que foram separadas erroneamente. A alimentação deste dicionário ocorre de forma manual.

3.3.4 Identificador fonético

A etapa de identificação fonética realiza a transcrição fonética, que consiste em identificar os fonemas presentes em cada palavra. O identificador fonético possui um método de transcrição para cada classe de palavra, sendo o método `TranscreverClassePalavra` o mais importante. Neste método encontram-se as regras para identificação dos fonemas do alfabeto completo, incluindo os encontros consonantais, a acentuação etimológica e as regras do novo acordo ortográfico gramatical. Ainda é no identificador fonético que é feito a soletração de siglas e o processamento prosódico (explicado na próxima seção).

É necessário salientar que o identificador fonético insere pequenos trechos de silêncio quando achar necessário, entre a transcrição de palavras. O silêncio pode ser identificado pelo caractere "_" ou *underscore* no resultado da transcrição. Este comportamento serve para evitar a associação de fonemas que são impronunciáveis e que podem gerar erros durante a síntese feita pelo MBROLA.

Há também o caso das palavras que possuem as mesmas vogais como núcleo da sílaba, sem acentuação, mas que são pronunciadas diferentemente. Cita-se como exemplo as palavras *pele* e *dele*, pronunciadas como "*péle*" e "*dêle*". Sem a acentuação, o protótipo não sabe diferenciar a pronuncia do fonema /e/ em cada palavra. Este problema deve-se aos hábitos fonéticos diferenciados existentes na língua portuguesa, que determina a sílaba tônica de uma palavra baseado em sua origem etimológica.

> Dentro da nossa língua, é impossível enfeixar as nossas palavras em regras especiais que determinem o seu acento. Em regra geral, êste [sic] é dado aos nossos vocábulos de conformidade com acento que têm na língua de origem e não de acordo com a terminação ou qualquer característico no próprio português. Assim, se dizemos que *Gibraltar é oxítono*, e *Amílcar paroxítono* (palavras de idêntica terminação) é por ser êsse [sic] o acento etimológico. (ALMEIDA, 1969, p. 54)

Para minimizar estes problemas, algumas regras foram adicionadas ao identificador fonético, criando um comportamento padrão para a pronúncia de certas palavras que não

empregam acentuação. As palavras que não seguirem o comportamento padrão, devem ser adicionadas ao dicionário de exceções.

3.3.4.1 Encontros consonantais

Os encontros consonantais inseparáveis cuja segunda consoante é ***l*** ou ***r***, não necessitam de tratamento, porém, por limitações do sintetizador alguns encontros consonantais são impronunciáveis e ocasionam erros irrecuperáveis durante a síntese, como por exemplo nas palavras *gnomo*, *apto*, *digno*, *mnemônico* e *ritmo*. Para estes casos, Cunha e Cintra (1985, p. 51) explicam que "na linguagem coloquial brasileira há, [...] uma acentuada tendência de destruir estes encontros de difícil pronúncia pela intercalação da vogal *i* (ou *e*)". Esta tendência se mostrou a solução para os problemas destes encontros consonantais. Por este motivo, as letras ***b***, ***d***, ***p***, ***t*** e ***f***, cuja transcrição poderia ser direta, precisaram receber um tratamento especial no identificador fonético. O quadro 18 mostra o tratamento do fonema /f/, onde foi implementado um código que adiciona o fonema /i/, entre possíveis encontros consonantais problemáticos.

```
if letra = 'f' then begin
  vTranscricaoPalavra.Add(MontaFonemaComTempo('f')+' 50 90');
  //-- Resolve encontros consonantais, como em software, afta
  if pos(vPalavra[posletra+1],CONSOANTES_semRL) > 0 then begin
    vTranscricaoPalavra.Add(MontaFonemaComTempo('imudo'));
  end;
  inc(posLetra);
end;
```

Quadro 18 – Regra de transcrição do fonema /f/

Ainda existe o caso de palavras compostas cuja primeira palavra finaliza com consoante, como nas palavras *sub-bibliotecário*, *bem-nascido* e *super-homem*. O hífen não tem função fonética e, por isso, é um caractere ignorado pelo identificador fonético, o que resulta em uma nova construção de encontro consonantal que também é tratada.

3.3.4.2 Acentuação etimológica

O identificador fonético foi munido de alguns comportamentos padrões para alguns tipos de palavras em que não se emprega acentuação, mas que possuem pronúncia diferente da escrita. Estas regras foram criadas baseadas em testes realizados com diversos textos, onde se percebeu um padrão predominante. As regras podem ser listadas da seguinte maneira:

a) palavras com duas sílabas em que a vogal ***e*** é núcleo serão transcritas com o primeiro ***e*** aberto (*p**e**le*, *f**e**re*, ***e**rre*);
b) palavras com duas sílabas em que a vogal ***o*** é núcleo da primeira sílaba e a vogal ***a*** é núcleo da segunda sílaba, serão transcritas com o a vogal ***o*** aberta, exceto se a sílaba seguinte for ***lha*** ou seguida do encontro consonantal ***rr*** (*b**o**la*, *c**o**la*, *r**o**da*, *p**o**rta*);
c) palavras com uma sílaba com as vogais ***o*** ou ***e*** como núcleo, seguidas da sílaba ***que***, serão transcritas com a vogal aberta (*t**o**que*, *c**o**que*, *enf**o**que*, *x**e**que*, *l**e**que*, *mol**e**que*);
d) palavras com duas sílabas, com a vogal ***o*** como núcleo da primeira sílaba e a vogal ***e*** como núcleo da segunda sílaba, serão transcritas com a vogal ***o*** aberta (*p**o**te*, *p**o**rre*, *j**o**gue*).

Observa-se que três destas regras se limitam às palavras de duas sílabas. Isso se deve ao fato de que a maioria das palavras com mais de duas sílabas empregam acentuação quando modificam a tonicidade da vogal, caso contrário, todas as vogais, exceto ***a***, são pronunciadas fechadas. Ainda assim, algumas palavras que se encaixam nas regras descritas têm sua transcrição inadequada, como por exemplo, as palavras *b**o**ca* e *t**o**uca*. As duas palavras se encaixam na segunda regra, porém sua pronúncia é com a vogal ***o*** fechada. Estas palavras devem ser adicionadas ao dicionário de exceções para que sejam transcritas corretamente.

O quadro 19 apresenta o código de transcrição da vogal ***e***, sendo que nas linhas 14 a 24 é possível verificar a implementação das regras descritas nos itens (a) e (c).

```
01  //-- Regra de transcrição dos fonemas /e/, /ee/ e /em/
02  if pos(letra,'eêé') > 0 then begin
03    vSilaba := BuscarSilaba;
04    vPosNaSilaba := pos(letra,vSilaba);
05    //-- Verfica se é vogal nasal (dígrafo EM ou EN)
06    if (vSilaba[vPosNaSilaba+1] = 'm') or (vSilaba[vPosNaSilaba+1] = 'n') then
07    begin
08      vTranscricaoPalavra.Add(MontaFonemaComTempo('em'));
09      //-- Consome a proxima letra (que é o M ou N)
10      inc(posLetra);
11    end
12    else begin
13      if (letra = 'é') or
14         //-- Acompanhada da sílaba QUE, como em tOQUE, rOQUE, enfOQUE
15         ((pos(letra,VOGAL_e) > 0) and
16          (vPalavra[posLetra+1] = 'q') and
17          (vPalavra[posLetra+2] = 'u') and
18          (pos(vPalavra[posLetra+3],VOGAL_e) > 0)) or
19         //-- Acompanhada de uma sílaba final com E, como em pEguE, lEquE
20         ((letra = 'e') and
21          (prToken.getQuantidadeSilabas = 2) and
22          ((vPalavra[Length(vPalavra)] = 'e') or
23          (Copy(vPalavra,Length(vPalavra)-1,2) = 'es')) and
24          (posLetra <> Length(vPalavra))) or
25         //-- Regra do novo acordo gramatical
26         //-- Se estiver na sequencia "eia" ou "eio" no final da palavra,
27         //-- como em alcatEIA, estrEIA, colmEIA, corEIA, estrEIO
28         ((vPalavra[posLetra]   = 'e') and
29          (vPalavra[posLetra+1] = 'i') and
30          (pos(vPalavra[posLetra+2],'ao') > 0) and
31          ((posLetra+2 = Length(vPalavra)) or
32          (vPalavra[posLetra+3] = 's'))) then begin
33        vTranscricaoPalavra.Add(MontaFonemaComTempo('ee'));
34      end
35      else begin
36        //senão usa-se na forma natural, vogal fechada E
37        vTranscricaoPalavra.Add(MontaFonemaComTempo('e'));
38      end;
39    end;
40    inc(posLetra);
41  end;
```

Quadro 19 – Regra de transcrição para a vogal ***e***

3.3.4.3 Regras do novo acordo ortográfico

O novo acordo ortográfico, assinado em 1990, que entrou em vigor no ano de 2009, ficará em transição até 2012, que será o ano em que as novas regras ortográficas serão efetivamente oficializadas (CONFEDERAÇÃO NACIONAL DOS TRABALHADORES EM EDUCAÇÃO, 2009). As modificações ortográficas ratificadas no acordo dificultaram ainda mais alguns casos de identidade fonética de certos vocábulos da língua portuguesa. Contudo, a maior vantagem deste novo acordo, é que um sistema texto-fala desenvolvido para o Brasil, pode naturalmente transcrever um texto escrito em outro país, cujo idioma oficial seja o português, pois a ortografia será a mesma. O que muda nestes casos são os hábitos fonéticos

de cada nação.

O novo acordo propõe mudanças em várias partes da língua portuguesa, porém colide apenas em quatro momentos com a tarefa de transcrição:

a) a inclusão das letras ***k***, ***w*** e ***y*** no alfabeto oficial;

b) a abolição do trema;

c) o abandono do acento agudo nos ditongos ***éi*** e ***ói*** das palavras paroxítonas;

d) o emprego do hífen (ou a falta dele) em algumas palavras compostas.

Para as letras ***k***, ***w*** e ***y***, o identificador fonético utiliza os respectivos fonemas, /k/, /u/ e /i/, para realizar a transcrição. Portanto, a palavra *show*, que é uma palavra estrangeira, porém incorporada ao dicionário brasileiro, seria transcrita conforme o quadro 20.

```
x 100
o 90
w 95
_ 5 10 90
```

Quadro 20 – Transcrição da palavra *show*

O segundo ponto impactante do novo acordo ortográfico é a abolição do trema. "Emprega-se o trema no *u* cuja pronúncia se percebe depois de *q* ou *g*, antes de *e* ou *i*" (BARBOSA, 1973, p. 21). Neste caso "coloca-se o trema [...] nos grupos *gue*, *gui*, *que*, *qui*, quando proferido átono" (CEGALLA, 1985, p. 75). Com o novo acordo, as palavras em que havia o emprego do trema, perdem sua identidade fonética. A pronúncia destas palavras fica a mercê dos hábitos fonéticos do leitor.

Para manter a naturalidade da pronúncia destas palavras, o identificador fonético precisou aprovisionar novas regras para os fonemas /g/ e /q/. O quadro 21 apresenta o trecho de código completo da transcrição da letra ***q***, onde se observa um exemplo de duas regras criadas para tratamento da ausência do trema. Nas linhas 7 a 11 do quadro 21, é implementada a regra para palavras com a sílaba ***quen***, em que havia o emprego do trema. Conforme o novo acordo ortográfico, estas palavras perdem o uso do trema, mas continuam a ser pronunciadas como ***qwen***. A regra criada para estes casos pode ser descrita da seguinte maneira. Se a letra ***q*** estiver acompanha da vogal ***u***, sucedida pela vogal ***e***, mais a consoante ***n*** e a letra ***q*** não é a primeira letra da palavra, o identificador fonético pronuncia a vogal ***u***, caso contrário, permanece a regra das sílabas ***que***. Alguns exemplos de palavras que podem ser tratadas por esta regra são: *cinquenta*, *sequencia*, *frequente* e *delinquente*.

Nas linhas 13 a 17, do quadro 21, é implementada uma regra para as palavras com a sílaba ***ques*** em que havia o emprego do trema, como por exemplo, nas palavras *equestre* e *sequestro*.

```
01  if (letra = 'q') then begin
02    if ((vPalavra[posLetra+1] = 'ü') and
03        (pos(vPalavra[posLetra+2],VOGAIS) > 0)) or
04       //-- Se o Q aparecer nas sílabas QUA ou QUO, deve-se pronunciar o U
05       ((vPalavra[posLetra+1] = 'u') and
06        (pos(vPalavra[posLetra+2],VOGAL_a+VOGAL_o) > 0)) or
07       //-- Regra para o novo acordo gramatical (sem trema)
08       ((pos(vPalavra[posLetra+1],'uü') > 0) and
09        (pos(vPalavra[posLetra+2],VOGAL_e) > 0) and
10        (vPalavra[posLetra+3] = 'n') and
11        (posLetra > 1)) or
12       //- Regra para o novo acordo gramatical (sem trema)
13       ((pos(vPalavra[posLetra+1],'uü') > 0) and
14        (pos(vPalavra[posLetra+2],VOGAL_e) > 0) and
15        ((vPalavra[posLetra+3] = 's') and
16         (posLetra <> Length(vPalavra))) and
17        (posLetra > 1)) then begin
18      //-- Usa-se o fonema da letra C
19      vTranscricaoPalavra.Add(MontaFonemaComTempo('k'));
20    end
21    else begin
22      //-- senão, não deve-se pronunciar o som da vogal U
23      vTranscricaoPalavra.Add(MontaFonemaComTempo('k'));
24      //-- consome a letra U
25      inc(posLetra);
26    end;
27    inc(posLetra);
28  end;
```

Quadro 21 – Regra de transcrição para o fonema /q/

Ainda há o abandono do acento agudo nos ditongos ***éi*** e ***ói*** das palavras paroxítonas, como por exemplo, *estreia*, *colmeia*, *asteroide* e *celuloide*. No quadro 19, nas linhas 25 a 32, é possível verificar a implementação desta regra que pode ser descrita da seguinte forma: a letra ***e*** deve ser seguida da vogal ***i***, que por sua vez, deve ser seguida por ***a*** ou ***o***, quando estas duas últimas vogais estiverem no final da palavra ou a próxima letra dessas vogais for a consoante ***s***. O mesmo tipo de regra vale para o ditongo ***ói***, implementado na transcrição da vogal ***o***.

O hífen é um sinal de pontuação sem função fonética e por isso sempre é ignorado pelo identificador fonético. Algumas palavras compostas na língua portuguesa perderam o emprego do hífen e estas palavras geralmente remontam encontros vocálicos e consonantais pronunciáveis. Alguns exemplos de palavras concatenadas pelo novo acordo ortográfico são: *autoescola*, *intermunicipal*, *semirreta*, *coeducar*. Por este motivo, o identificador fonético não apresenta nenhum tratamento diferenciado para esta alteração do uso do hífen.

Vale acrescentar que todas estas regras implementadas para tratamento do novo acordo ortográfico foram criadas de forma arbitrária, baseadas em uma análise por amostragem, onde foi possível identificar uma tendência ou padrão de escrita. É possível que existam palavras que não se encaixem nas regras implementadas, o que resultará numa transcrição inadequada, porém, ortograficamente coerente.

3.3.4.4 Soletração de siglas

As palavras classificadas como siglas são transcritas e pronunciadas pelo protótipo letra por letra. O método `TranscreverClasseSigla`, no objeto de identificação fonética, é o local onde está implementada a transcrição por extenso de cada letra do alfabeto (incluindo as letras do novo acordo ortográfico). A transcrição de cada letra é feita conforme sua pronúncia. Assim, por exemplo, a letra ***w*** cuja pronúncia é *dábliu*, teve sua transcrição obtida informando a palavra *dábliu*, conforme pode ser visto no quadro 22, que apresenta adicionalmente o código de transcrição da letra *r*.

```
Ord('W'): begin
  vTranscricaoPalavra.Add('d '+IntToStr(tempoD+vTempoExtra));
  vTranscricaoPalavra.Add('a '+IntToStr(tempoA+vTempoExtra));
  vTranscricaoPalavra.Add('b '+IntToStr(tempoB+vTempoExtra));
  vTranscricaoPalavra.Add('l '+IntToStr(tempoL+vTempoExtra));
  vTranscricaoPalavra.Add('y '+IntToStr(tempoY+vTempoExtra));
  vTranscricaoPalavra.Add('u '+IntToStr(tempoU+vTempoExtra));
end;

Ord('R'): begin
   vTranscricaoPalavra.Add('ee '+IntToStr(tempoEe+vTempoExtra));
   vTranscricaoPalavra.Add('r2 '+IntToStr(tempoR2+vTempoExtra));
   vTranscricaoPalavra.Add('r '+IntToStr(tempoR+vTempoExtra));
   vTranscricaoPalavra.Add('e '+IntToStr(tempoE+vTempoExtra));
end;
```

Quadro 22 – Transcrição da soletração das letras *w* e *r*

Existe, porém, uma limitação nesta estratégia adotada de soletração. Algumas siglas usuais formam palavras pronunciáveis, que por hábito não são soletradas, como por exemplo, as siglas FURB e UNICAMP. Nestes casos o protótipo não faz a pronúncia corrida destas siglas, mas sim a soletração, conforme visto no exemplo do quadro 23.

```
u 100
_ 30
e 120
n 110
e 120
i 140
_ 30
s 120
e 120
a 160
_ 10
e 120
m 110
e 120
p 140
e 120
_ 5 10 90
```

Quadro 23 – Transcrição da sigla UNICAMP

3.3.5 Processador prosódico

Para o protótipo desenvolvido, o processador prosódico apresenta-se implementado junto ao identificador fonético, uma vez que os sinais de pontuação são tratados e classificados pelo analisador léxico. A prosódia nada mais é que a "pronúncia regular das palavras com a devida acentuação" (FERREIRA, 1975, p. 1148).

O identificador fonético possui o método `TranscreverClassePontuacao`, que é responsável pela transcrição prosódica do texto. Com exceção da interrogação, da exclamação e da reticência, os demais caracteres de pontuação (vírgula, ponto e vírgula, dois pontos, travessão, ponto final) não recebem nenhum tratamento especial de acentuação da pronúncia, eles consistem apenas em adicionar um silêncio após a palavra.

O quadro 24 demonstra a transcrição da frase "Venha, temos muito o que conversar." que utiliza vírgula e ponto final. Observa-se que após a transcrição da palavra *Venha* é acrescentado um silêncio adicional de 110 milissegundos devido ao uso da vírgula. Ao final da frase, também é adicionado outro silêncio ainda maior, de 330 milissegundos devido uso do ponto final.

```
v 60 50 90
e 90
nh 90
a 100 10 86 90 82
_ 5 10 90
_ 110 10 90
t 100 50 90
e 90
m 80
o 90
s2 90
_ 5 10 90
m 80
u 40
y 80
t 100 50 90
o 90
_ 5 10 90
o 130
k 105
e 90
_ 5 10 90
k 105
om 90
v 60 50 90
e 90
r2 60
s 90
a 100 10 86 90 82
r2 60
_ 5 10 90
_ 330 10 90
```

Quadro 24 – Transcrição da frase "Venha, temos muito o que conversar."

Para realizar o processamento prosódico de uma frase interrogativa, foi elaborada a implementação apresentada no quadro 25.

```
01  if FTranscricaoCompleta.Count > 7 then begin
02    //-- Busca a posição que deve receber pitch de 100% (7 fonemas antes)
03    //-- deduz 8 porque começa do zero
04    vPosNormalizadora := FTranscricaoCompleta.Count-8;
05    //-- Busca o fonema
06    vFonema := FTranscricaoCompleta[vPosNormalizadora];
07    //-- Adiciona um pitch de 100% para não alterar o início da frase
08    vFonema := AdicionarPitch(vFonema,[50],[100]);
09    //-- Coloca o fonema com o pitch alterado
10    FTranscricaoCompleta[vPosNormalizadora] := vFonema;
11  end;
12  //-- Busca a posição que deve receber pitch de 180% (ultimo fonema)
13  vPosInterrogacao := ifthen(((FTranscricaoCompleta.Count-2) >= 0),
                               FTranscricaoCompleta.Count-2,0);
14  //-- Busca o fonema
15  vFonema := FTranscricaoCompleta[vPosInterrogacao];
16  //-- Adiciona um pitch de 180% para dar sensação de interrogação
17  vFonema := AdicionarPitch(vFonema,[10],[180]);
18  //-- Coloca o fonema com o pitch alterado
19  FTranscricaoCompleta[vPosInterrogacao] := vFonema;
20  //-- Aumenta a duração do ultimo fonema vogal
21  vUltVogal := 0;
22  //-- Busca a última vogal transcrita
23  repeat
24    inc(vUltVogal);
25    vFonema := FTranscricaoCompleta[FTranscricaoCompleta.Count - vUltVogal]
26  until (pos(vFonema[1],VOGAIS+'@'+'yw') > 0);
27  //-- Altera o tempo para 200 milisegundos
28  vFonema := AlterarTempo(vFonema,200);
29  FTranscricaoCompleta[FTranscricaoCompleta.Count - vUltVogal] := vFonema;
30  //-- Adiciona um silêncio para normalizar o pitch
31  FTranscricaoCompleta.Add('_ 400 10 100');
```

Quadro 25 – Código de processamento prosódico da interrogação

No sétimo fonema antes da posição da interrogação, altera-se a frequência para 100 Hz (linha 02 a 10), e no fonema imediatamente anterior ao final da interrogativa, aumenta-se a frequência para 180 Hz (linha 12 a 19). Adicionalmente à alteração na frequência, faz-se o aumento do tempo da última vogal transcrita para 200 milissegundos (linha 20 a 29). O resultado deste processamento é uma pronúncia similar a uma pergunta. O quadro 26 apresenta um exemplo de transcrição com processamento prosódico interrogativo da frase “Bom dia senhor?”.

```
b 105 50 90
om 90
_ 5 10 90
d 100 50 90
i 80
a 100 50 100
_ 5 10 90
s 90
e 90
nh 90
o 200
r2 60 10 180
_ 5 10 90
_ 400 10 100
```

Quadro 26 – Transcrição da frase interrogativa “Bom dia senhor?”

O processamento prosódico da exclamação e da reticência possui um código semelhante à interrogação. Porém, o comportamento da exclamação reduz o tempo dos fonemas ao final da frase e aumenta a frequência para 110 Hz. A reticência diminui a frequência e prolonga o tempo da última vogal.

3.3.6 Operacionalidade da implementação

A operacionalidade do protótipo é bastante simplificada. É importante salientar que o mesmo não está preparado para ser operado por pessoas com deficiência visual severa (que não enxergam). Isso se deve ao fato de que o objetivo principal da interface é testar a qualidade do processamento textual que é realizado inteiramente pelo pacote (biblioteca) desenvolvido com nome FurbTTS.

Como estudo de caso é realizado o processamento da seguinte frase extraída do *site* do Departamento de Sistemas e Computação (DSC) da Universidade Regional de Blumenau (VAHLDICK, 2009): “Nesse último fim de semana aconteceu a Final Brasileira da Maratona de Programação, em Campinas.”

Ao executar o protótipo é apresentada ao usuário a tela da figura 27. Inicialmente o usuário não pode utilizar nenhum recurso de correção de resultados. Ele pode somente consultar o dicionário silábico (figura 27(1)), consultar o dicionário de exceções (figura 27(2)) e informar algum texto de entrada para processamento no primeiro campo nomeado `Entrada de texto` (figura 27(3)).

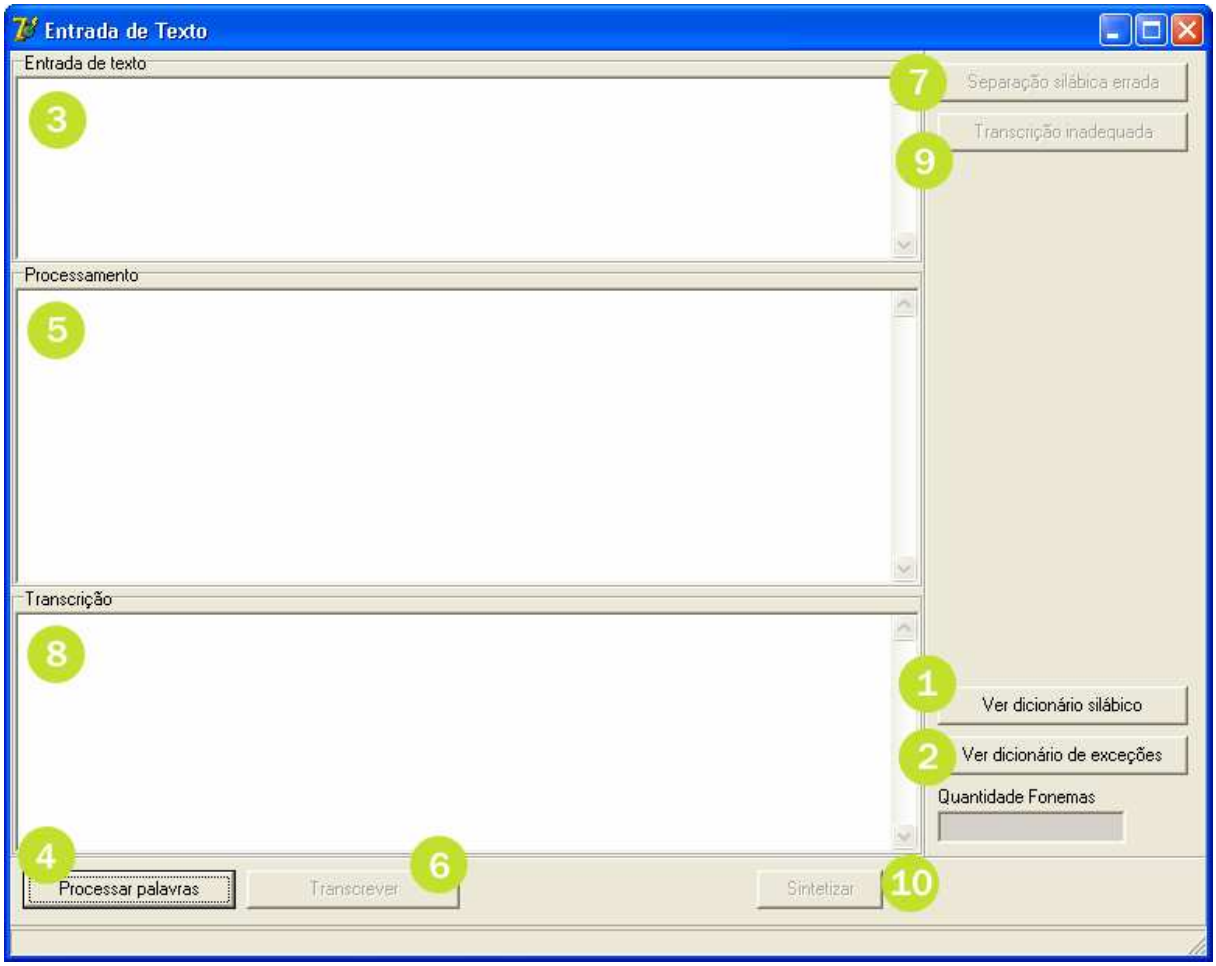

Figura 27 – Interface do protótipo

Para iniciar o processamento, o usuário precisa digitar ou colar no primeiro campo da tela um texto em português. Não são permitidos textos com marcadores (itens), textos com parênteses, aspas ou colchetes não agrupados, ou textos com múltiplas pontuações seguidas (exceto reticência). Esta é uma prática necessária para que o texto inserido não cause erros no analisador sintático.

Após a entrada do texto, o usuário deve clicar no botão `Processar palavras` (figura 27(4)), para que o programa faça a primeira etapa do processamento. O resultado desta ação está representado na figura 28. As palavras, classificadas e separadas silabicamente são apresentadas no segundo campo, denominado `Processamento` (figura 27(5)).

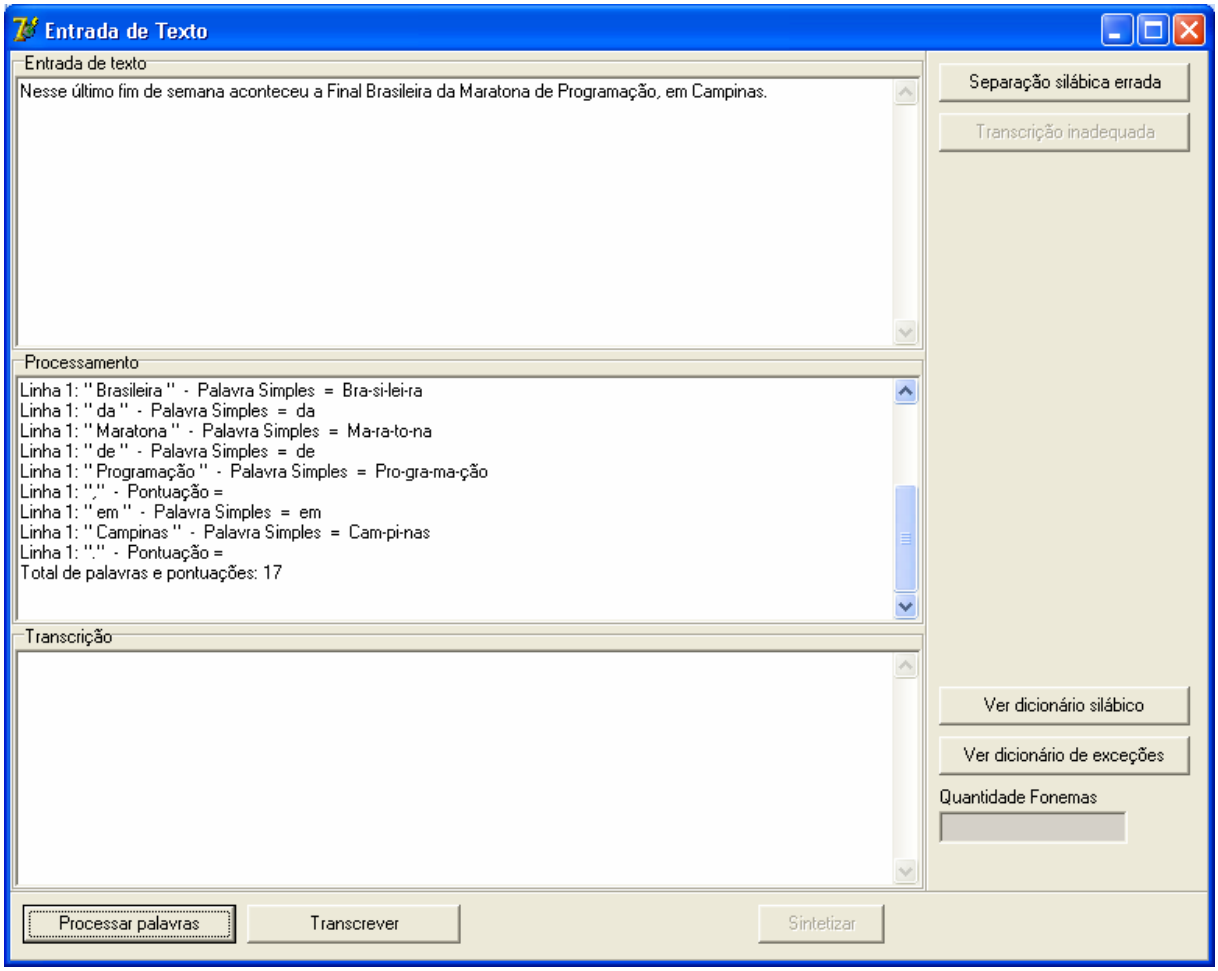

Figura 28 – Processamento das palavras

O usuário pode então analisar o resultado da classificação e a separação silábica de cada palavra. Com isso são habilitados os botões `Transcrever` (figura 27(6)) e `Separação silábica errada` (figura 27(7)). Caso o usuário detecte uma separação silábica incorreta, deve marcar a palavra no texto no campo `Entrada de texto` e clicar no botão `Separação silábica errada`. O protótipo apresenta uma nova tela com a palavra selecionada conforme demonstrado na figura 29, com a palavra *aconteceu* encontrada na frase do estudo de caso.

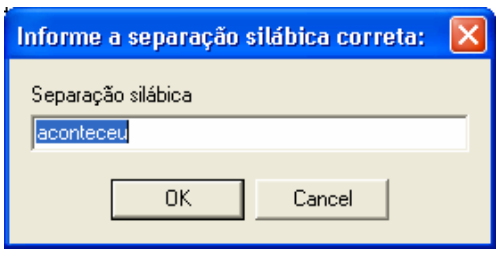

Figura 29 – Tela para correção silábica

Nesta tela o usuário deve informar na caixa de texto, a separação silábica correta da palavra, separando as sílabas por hífen, conforme assinala a norma da gramática da língua portuguesa. Após informar a separação, o protótipo emite uma mensagem de confirmação com a separação informada. Caso seja confirmada a separação, esta palavra é adicionada ao

dicionário silábico. Se não for confirmada, o protótipo não realiza nenhuma tarefa.

Neste ponto, o usuário pode clicar novamente no botão `Processar palavras`, para que as palavras que foram adicionadas ao dicionário silábico sejam processadas corretamente, ou pode clicar no botão `Transcrever para` o identificador fonético seja acionado.

Caso o botão `Transcrever` seja pressionado, é realizada a transcrição das palavras processadas e são habilitados os botões `Transcrição inadequada` (figura 27(9)) e `Sintetizar` (figura 27(10)). A transcrição completa do texto é apresentada no terceiro campo da tela, chamado `Transcrição` (figura 27(8)), contendo a lista de fonemas, os seus tempos e a informação prosódica. O resultado do processamento e da transcrição está representado na figura 30 abaixo.

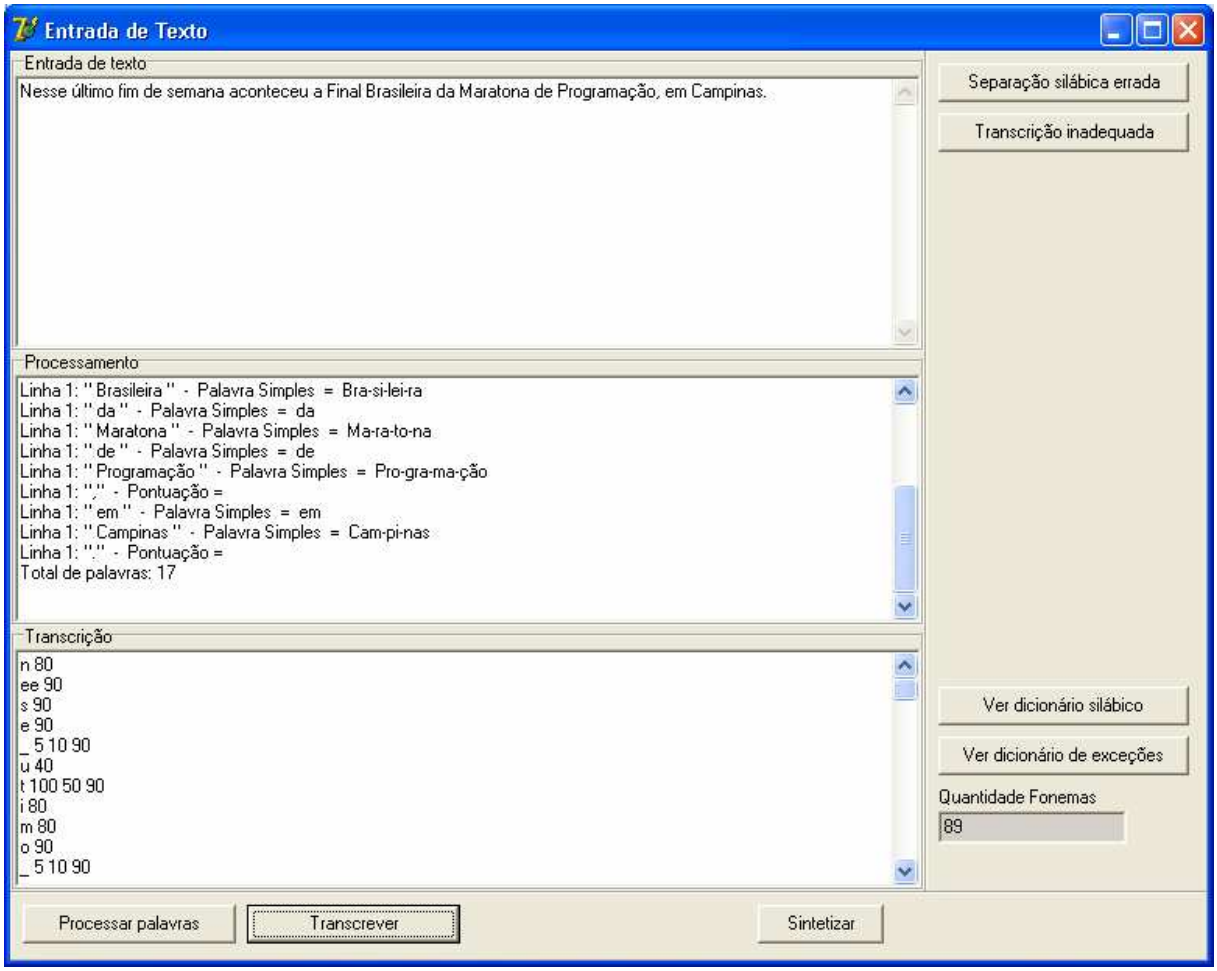

Figura 30 – Transcrição do texto e processamento das palavras

Paralelamente a transcrição, o protótipo gera um arquivo texto de saída nomeado `Transcricao.pho` com as mesmas informações de transcrição e no mesmo diretório onde encontra-se o executável do programa.

O usuário então pode escolher ouvir a transcrição ou corrigir uma transcrição inadequada de alguma palavra. Caso decida-se por corrigir uma transcrição inadequada, deve

selecionar a palavra no primeiro campo e clicar no botão `Transcrição inadequada`. Então o protótipo apresenta a tela da figura 31, que no caso contém a palavra *Maratona* selecionada no texto de entrada, juntamente com a sua transcrição atual. O usuário pode modificar todos os aspectos da transcrição desta palavra, porém, deve atentar para o formato de saída de transcrição, pois o protótipo não valida a estrutura da informação inserida pelo usuário. Para confirmar a nova transcrição, o usuário deve clicar em `Gravar no dicionário` e para cancelar, basta fechar a janela para que nenhuma tarefa seja realizada. Se a alteração for confirmada, a nova transcrição é armazenada no dicionário de exceções.

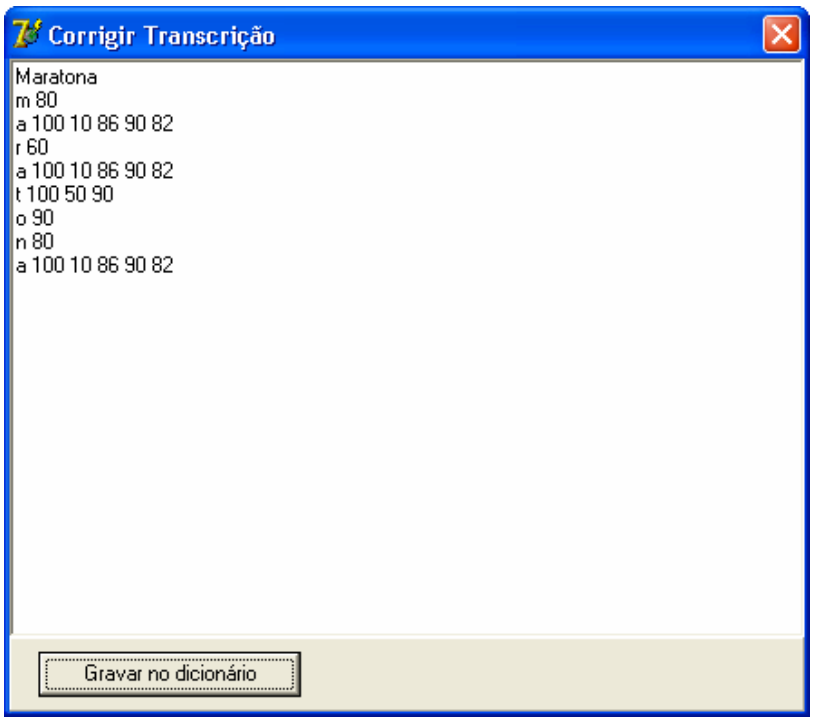

Figura 31 – Tela para correção de transcrição inadequada

Após corrigir uma transcrição inadequada, o usuário pode realizar a transcrição novamente, clicando no botão `Transcrever`, agora usando a transcrição correta armazenada no dicionário de exceções. O botão `Sintetizar` aciona o sintetizador MBROLA (integrado ao protótipo) que gera um arquivo Wave para ser executado.

3.4 RESULTADOS E DISCUSSÃO

Os resultados encontrados com o término do trabalho são satisfatórios, visto que a maioria dos textos escritos em português inseridos no protótipo foram transcritos e também sintetizados sem problemas. Observa-se que, inicialmente, conforme proposto nos objetivos

específicos, o protótipo deveria processar um vocabulário restrito. No entanto, o protótipo foi implementado para permitir o processamento de qualquer texto corretamente escrito (léxica e sintaticamente) na língua portuguesa, com desempenho satisfatório e principalmente inteligível.

No quadro 32 é apresentado um comparativo das características dos trabalhos correlatos com o protótipo implementado. No primeiro item pode-se observar que todos os trabalhos, exceto o de Hubner (1992), processam um vocabulário irrestrito. Salienta-se que o trabalho desenvolvido por Hubner (1992) não é um sistema texto-fala, mas um processador de linguagem natural.

Quanto à aplicação de IA no desenvolvimento, o protótipo (FurbTTS) é o único que não faz uso de nenhum conceito relacionado. O trabalho de Gomes (1998) recebe pontuação neste quesito, pois faz uso de IA somente no sintetizador acústico, que também é uma característica que só esse trabalho possui. Em contrapartida, embora seja o trabalho mais recente, o protótipo FurbTTS é o único que está habilitado para transcrição de textos com o novo acordo ortográfico.

	FurbTTS	BENTO (1995)	GOMES (1998)	HUBNER (1992)
vocabulário irrestrito	✓	✓	✓	⊘
conversão para uma linguagem formal	✓	✓	✓	✓
aplicação de inteligência artificial	⊘	✓	✓	✓
síntese de texto	✓	⊘	✓	⊘
sintetizador acústico próprio	⊘	⊘	✓	⊘
dicionário de exceções	✓	⊘	✓	⊘
dicionário silábico	✓	⊘	⊘	⊘
passível de extensão	✓	✓	✓	✓
suporte ao novo acordo ortográfico	✓	⊘	⊘	⊘

Quadro 32 – Características do FurbTTS com trabalhos correlatos

Adicionalmente, no dia 19 de novembro, foi realizada na sede da ACEVALI[8] uma demonstração do software com o intuito de avaliar a qualidade da transcrição. Nesta demonstração estavam presentes duas pessoas com deficiência visual (sem visão), João Agostinho Ângelo (atleta de Goabol) e Luana Tillmann (estudante e instrutora de ensino), e a diretora da instituição, a Sra. Vera.

[8] É uma entidade sem fins lucrativos, localizada em Blumenau, que oferece tanto qualificação para deficientes visuais, quanto assistência social, de saúde e previdenciária.

A demonstração consistiu na síntese de dois textos coletados da internet. Após cada apresentação, seguiu-se uma entrevista com as pessoas presentes, de modo que pudessem expressar sua opinião e sugestões sobre o estágio de desenvolvimento que se encontrava o software e, em especial, sobre pronúncia das palavras, ruídos da síntese e a informação prosódica (pontuação e entonação).

Na primeira apresentação (primeiro texto), foi indicado qual era o tema do texto, para que as pessoas presentes pudessem orientar seu raciocínio. Após a síntese completa, foi feita uma pergunta sobre o tema abordado, sendo necessário ter compreendido o texto para respondê-la. A pergunta foi corretamente respondida pelos presentes.

Na segunda apresentação (segundo texto), não foi indicado o tema, sendo que os entrevistados deveriam identificar o assunto. Após a síntese completa, os entrevistados identificaram corretamente o tema do segundo texto e ainda responderam uma pergunta sobre o assunto abordado, que apenas foi respondida pela entrevistada Luana, sendo que João não compreendeu a parte que respondia esta pergunta.

Todos os entrevistados afirmaram que a voz utilizada está ainda muito robótica, bastante distante da naturalidade de uma fala humana. Contudo, esta característica não impede a compreensão da fala, mas dificulta imensamente entender a síntese em um local com ruídos externos, o que demanda toda atenção do usuário. Em contrapartida, foi elogiada a pontuação e a informação prosódica, onde pode-se claramente ver a presença de vírgula, ponto final, interrogação, entre outros.

Por fim, foi apontado que durante a apresentação do segundo texto, o entendimento foi melhorado. De acordo com os entrevistados, o cérebro acostuma-se com a fala estranha e começa a identificar os vícios da síntese, ignorando os erros de pronúncia.

4 CONCLUSÕES

Apesar de ser alvo de estudos há quase 20 anos, as tecnologias referentes à linguagem natural computacional só se tornaram parte presente do cotidiano em aplicações que se limitam quase sempre a atender um objetivo bem específico dentro de um universo definido. Isto se deve a constante complexidade de efetuar o processamento de linguagem natural. A exemplo disso, pode-se citar aplicações como os aparelhos celulares com reconhecimento de marcas vocais, os sistemas automatizados de tele-atendimento e até mesmo carros com sistemas incorporados para leitura de mensagens SMS.

Neste trabalho foi analisada a questão do processamento lingüístico. Também foram usados conceitos de orientação a objetos para implementação de quatro etapas do processo de conversão texto-fala (pré-processamento, análise linguística, transcrição fonética, processamento prosódico).

Inicialmente foi proposto que o protótipo deveria processar adequadamente textos escritos em português de um vocabulário restrito. O protótipo implementado supera este objetivo, pois não impõe restrições ao vocabulário processado. Também foi definido que o processamento linguístico efetuado pelo protótipo deveria comportar regras gramaticais da língua portuguesa, incluindo o novo acordo ortográfico. Verifica-se que o protótipo implementa várias normas gramaticais (incluindo uma verificação sintática) e também as regras do novo acordo ortográfico. Ao final do processo é gerado um arquivo com a transcrição completa, que inclusive incorpora um sintetizador para verificação do resultado. Todo o processamento linguístico é realizado em um pacote (biblioteca), que pode ser incorporado a qualquer outro projeto desenvolvido em Delphi. Com isso, conclui-se que todos os objetivos foram alcançados, embora com limitações.

A maior dificuldade dentro de todo o trabalho foi pesquisar as regras gramaticais que norteiam a estrutura fonética do idioma nacional. Em muitos casos simplesmente não existe norma e nem regra para definir a pronúncia da palavra. As gramáticas justificam a pronúncia baseada nos hábitos fonéticos da população ou na sua palavra de origem (normalmente do latim ou do grego). Desta forma, foi necessário um levantamento bibliográfico muito mais demorado do que o esperado. Muitas das regras definidas e implementadas foram arbitrariamente geradas a partir de uma análise feita em cima de amostras do vocabulário. Por isso, mesmo sendo alvo de estudos a quase 20 anos, um sistema texto-fala sempre será um desafio, pois existem muitas particularidades fonéticas não-metódicas, que impedem o avanço

de um sistema sintetizador livre de uma base de dados ou dicionário de exceções.

O sintetizador MBROLA, aliado ao GALS, foi de altíssima importância para o desenvolvimento deste protótipo. As duas ferramentas nortearam todo o desenvolvimento. O GALS através da geração dos analisadores léxico e sintático, reduziu em muito o trabalho para pré-processamento do texto de entrada, e o sintetizador MBROLA orientou precisamente como seria o arquivo de saída, possibilitando validar a sua qualidade fonética. O ambiente de programação Delphi mostrou-se adequado para tratamento de cadeias de caracteres, sendo que os componentes nativos foram suficientes para todas as necessidades de construção do protótipo.

Espera-se que o protótipo desenvolvido possa contribuir no desenvolvimento de uma ferramenta completa de *text-to-speech,* que possivelmente possa ser incorporada ao *site* de instituições de ensino e dê a possibilidade para pessoas com deficiência visual acessar o conteúdo destas instituições.

Adicionalmente, o protótipo ainda possui muitas limitações no que diz respeito ao texto de entrada. Sabe-se que os textos de internet não são perfeitamente elaborados ou diagramados, havendo erros ortográficos, sintáticos e semânticos diversos, além do uso de palavras estrangeiras. O texto de entrada para o protótipo deve estar em perfeita conformidade com a gramática da língua portuguesa e com a gramática especificada no GALS, sob o risco de gerar erros que impeçam o funcionamento do programa.

Outra limitação é com relação a nomes próprios. Muitas vezes estas palavras apresentam sequências inválidas de letras se comparado com as normas da língua portuguesa. O nome do autor deste trabalho, por exemplo, apresenta um encontro consonantal impróprio para o idioma nacional (***th***). Estes tipos de palavras podem gerar erros durante o processamento linguístico ou durante a síntese e não permitem que o protótipo funcione adequadamente.

Pode-se salientar também a limitação do protótipo para gerar transcrições diferentes de uma mesma palavra, conforme explicado na seção 3.3.3. Esta característica negativa (oriunda da estratégia utilizada na análise linguística) pode prejudicar em alguns casos a compreensão imediata da síntese do texto.

Apesar destas limitações, o protótipo implementado teve um desempenho que surpreendeu as 15 das pessoas a qual ele foi apresentado. Algumas destas pessoas chegaram a solicitar cópias do programa, mesmo em estado incompleto, para usarem como um leitor autônomo de documentos. Em um dos testes realizados, foi escrita uma mensagem completa utilizando o protótipo e enviado por *e-mail* o arquivo de som (com duração aproximada de 5

minutos) ao destinatário, sem legendas e sem indicação de tema. O destinatário compreendeu 100% da mensagem e respondeu o *e-mail* dentro do tema abordado, inclusive, respondendo os questionamentos feitos, o que indica que o processamento prosódico mostrou-se perceptível.

Por fim, pode-se dizer que uma das vantagens mais importantes do protótipo também é a capacidade de tratar o novo acordo ortográfico, que entrou em vigor neste ano de 2009.

4.1 EXTENSÕES

Existem pontos que podem ser melhorados e incrementados, sendo eles:

a) efetuar um tratamento mais adequado do texto de entrada, ignorando palavras em outros idiomas, erros ortográficos e gramaticais (atualmente na presença de um erro no texto de entrada, não é possível continuar o processamento das palavras);
b) usar a biblioteca criada para o acesso, a captura e a sintetização de textos diretamente de *sites*;
c) refinar as regras implementadas, tratando a acentuação etimológica das palavras e os hábitos fonéticos da língua portuguesa, como por exemplo, nas palavras que possuem a mesma ortografia, mas com duas ou mais pronúncias. Neste caso, sugere-se aliar técnicas IA (raciocínio baseado em casos, redes de probabilidade) para análise de texto, conforme visto nos trabalhos correlatos;
d) desenvolver um módulo acústico totalmente novo, que utilize-se das informações de transcrição disponibilizadas pelo protótipo;
e) integrar a biblioteca gerada com outras plataformas de desenvolvimento ou outras linguagens.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, Napoleão M. **Gramática metódica da língua portuguesa**. 22. ed. São Paulo: Saraiva, 1969.

BARBOSA, Osmar. **Formulário ortográfico gramatical e noções elementares de gramática histórica**. São Paulo: Edigraf , [1973?].

BENTO, Douglas M. **Protótipo de um sintetizador de voz utilizando redes neurais**. 1995. 61 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

CEGALLA, Domingos P. **Novíssima gramática da língua portuguesa**. 26. ed. São Paulo: Companhia Editora Nacional, 1985.

_____. **Novíssima gramática da língua portuguesa**. 40. ed. São Paulo: Companhia Editora Nacional, 1997.

COSTA, Denis R. **BR3 database:** brazilian portuguese male. Release 2.021. [S.l.], 2005. Disponível em: <http://tcts.fpms.ac.be/synthesis>. Acesso em: 13 abr. 2009.

CONFEDERAÇÃO NACIONAL DOS TRABALHADORES EM EDUCAÇÃO.**Confira as regras de transição do acordo ortográfico**. [Brasília], 2009. Disponível em: <http://www.cnte.org.br/index.php?option=com_content&task=view&id=1509&Itemid=82>. Acesso em: 15 nov. 2009.

CUNHA, Celso; CINTRA, Luís F. L. **Nova gramática do português contemporâneo**. Rio de Janeiro: Nova Fronteira, 1985.

DEVMEDIA GROUP. **APS, .net, Java, Delphi, SQL e webdesign**. [S.l.], 2009. Disponível em: <http://www.devmedia.com.br/portal/default.asp>. Acesso em: 22 set. 2009.

DUTOIT, Thierry D. et al. **The MBROLA project**: towards a freely available multilingual speech synthesizer. [Bélgica], 2005. Disponível em: <http://tcts.fpms.ac.be/synthesis/mbrola/mbrola_entrypage.html>. Acesso em: 02 set. 2009.

FERREIRA, Aurelio B. H. **Novo dicionário da língua portuguesa**. Rio de Janeiro: Nova Fronteira, 1975.

FRANZEN, Evandro. **Estudo e implementação da programação genética para síntese de fala.** 2002. 113 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

GOMES, Leandro C. T. **Sistema de conversão texto-fala para a língua portuguesa utilizando a abordagem de síntese por regras.** 1998. 107 f. Dissertação (Mestrado em Engenharia Elétrica) – Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas.

HUBNER, Jomi F. **Interface em linguagem natural.** 1992. 54 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

KLATT, Dennis H. Software for a cascade/parallel formant synthesizer**. Journal of the Acoustical Society of America**, Melville, n. 67, p. 971-995, Mar. 1980.

LIMA, Rocha. **Gramática normativa da língua portuguesa**. 16. ed. Rio de Janeiro: José Olímpio Editora, 1991.

MACHADO, Lucas. Síntese de voz para produção de livros falados e inclusão social para deficientes visuais. In: FEIRA DE INICIAÇÃO CIENTÍFICA, 16., 2007, Porto Alegre. **Anais eletrônicos...** Porto Alegre: UFRGS, 2007. Disponível em: <http://ufrgsweb.ufrgs.br/node/247?page=1>. Acesso em: 16 set. 2009.

MANOSSO, Radamés. **Fonemas da língua portuguesa**. Curitiba, [set. 2008]. Disponível em: <http://www.radames.manosso.nom.br/gramatica/fonemas.htm>. Acesso em: 05 set. 2009.

MEIER, Cardy. **Números cardinais e ordinais**. [São Paulo], 2009. Disponível em: <http://www.profcardy.com/cardicas/cardinal.php>. Acesso em: 04 nov. 2009.

OSTERMANN FILHO, Paulo E. **Desenvolvimento de regras de pronúncia para síntese de fala em língua portuguesa.** 2002. 129 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

REBELO, Michel N. **Desenvolvimento de um protótipo de gerador de analisador léxico**. 2002. 59 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau. Disponível em: <http://www.bc.furb.br/docs/MO/2002/266424_1_1.pdf>. Acesso em: 17 set. 2009.

SIMÕES, Flávio O. **Implementação de um sistema de conversão texto-fala para o português do Brasil.** 1999. 185 f. Dissertação (Mestrado em Engenharia Elétrica) – Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas.

VAHLDICK, Adilson. **Final brasileira da maratona de programação**. [Blumenau], 2009. Disponível em: <http://www.inf.furb.br/index.php?acao=completa&cd=141>. Acesso em: 11 nov. 2009.

APÊNDICE A – Regras de transcrição dos fonemas suportados pelo protótipo

O quadro 33 apresenta as regras de transcrição de cada fonema.

fonema	regra de transcrição
/g/	O fonema /g/ deve ser aplicado à síntese da palavra quando sua pronúncia é palatal, como no caso das palavras: *guerra*, *cegonha*, *negócio*, *garoto*, etc. Explica-se que o ***g***, nesta pronúncia, é proveniente de um abrandamento etimológico do ***c*** duro ou do ***q*** latino (ALMEIDA, 1969, p. 41). Para isto, o fonema /g/ somente é utilizado dependendo da letra que o acompanha, conforme os seguintes casos: *ga* (***ga**ragem*), *gue* (***gue**to*), *gui* (***gui**rlanda*), *go* (***go**r**go**nzola*), *gu* (***gu**ri*), *gua* (***gua**raná*), *g+consoante* (*incó**gn**ita*). Quando a consoante ***g*** se apresentar seguida de ***e*** ou ***i***, deverá ser usado o fonema /j/.
/k/	O fonema /k/ também só deve ser transcrito quando sua pronúncia for palatal.Portanto, tem-se o fonema na ocorrência, das sílabas *ca* (***ca**sa*), *co* (*en**co**sto*), *cu* (***cu**mprimento*), *que* (*to**que***), *qui* (*li**qui**do*), quando em encontro consonantal (*pa**ct**o*) ou na presença da própria letra ***k***. Portanto, pode-se afirmar que o fonema /k/, exceto para a letra ***k***, depende sempre da letra que o acompanha.
/j/	Para representar o fonema /j/ existem duas letras: ***g*** e ***j***. De acordo com Cegalla (1997, p. 54), "grafa-se este ou aquele signo não de modo arbitrário, mas de acordo com a origem da palavra". No caso da síntese deste fonema, pode-se assumir seu uso, quando ocorrer a letra ***j***, que tem pronúncia constante, e quando na ocorrência da letra ***g***, seguida de ***e*** ou ***i***, como, por exemplo, nas palavras: *farin**ge**, **ge**sso*, ***gi**bi*, ***gi**lete*, *rabu**gi**ce*.
/s/ /s2/	A pronúncia da consoante ***s*** possui dois sons: sibilante forte e sibilante brando. O som sibilante forte corresponde ao seu som literal e alfabético. Contudo, quando o ***s*** assume som sibilante brando, tem som conseguintemente acidental correspondente ao do ***z*** (ALMEIDA, 1969, p. 44). Por isso deve-se usar o fonema /s/ somente quando a letra ***s*** não aparece entre duas vogais ou quando na ocorrência dos seguintes encontros consonantais: ***ss***, ***sc***, ***sç*** e a letra ***x*** seguido de uma consoante. Desta forma, o fonema /s/ deve ser usado nas palavras: *ace**ss**o*, *adole**sc**ente*, *cre**sç**o*, *e**xc**elente*, etc. Ainda existe o caso da letra ***x*** apresentar-se entre duas vogais, porém desempenhando som de ***s***. Para estes casos, em que a pronúncia segue uma origem etimológica, será utilizado o dicionário de exceções para uma transcrição correta. Por fim, a consoante ***c*** quando seguida da vogal ***e*** ou ***i*** (nas sílabas ***ce*** e ***ci***), pode ser transcrita para o fonema /s/. Devido a regras do sintetizador MBROLA, o fonema /s2/ substitui o fonema /s/ quando a consoante ***s*** aparecer ao final de uma palavra ou anteceder uma consoante, que não as letras ***c*** ou ***ç***. Exemplos do uso do fonema /s2/ ocorrem nas palavras *fiéi**s**, con**s**tante*, *e**s**cova*, *mai**s***.
/x/	O fonema /x/ deve ser aplicado à síntese da palavra, se e somente se, quando pronunciado com som chiante (som alfabético), como no caso das palavras: ***x**adrez*, *pra**x**e*, *bai**x**o*, *gra**x**o*. Quando o emprego letra ***x*** determinar outro som, que não o som alfabético, o identificador fonético deverá procurar outra regra, como para o caso das seguintes palavras: *máximo*, *sexto*, *sexo*, *exame*, *léxico*, *próximo*, *êxodo*, etc. Se estes casos não forem tratados por nenhuma outra regra fonética, caberá ao dicionário de exceções fazer a transcrição. O encontro consonantal ***ch*** também apresenta como característica o mesmo som chiante da letra ***x***, e por isso deve-se usar o fonema /x/ na sua transcrição.
/z/	Deve ser utilizado na transcrição da palavra, quando na ocorrência da letra ***z***, e também quando for encontrada a consoante ***s*** entre duas vogais. Ainda existe o caso da letra ***x*** representar o mesmo som sibilante brando da consoante ***z***. Estes casos serão tratados pelo dicionário de exceções, uma vez que a sua pronúncia diferenciada tem origem etimológica.
/m/ /n/	Os fonemas /m/ e /n/ somente serão utilizados na transcrição de uma palavra quando aparecerem no início de uma sílaba. Caso contrário, as letras ***m*** e ***n*** são simples sinais de

fonema	regra de transcrição
	nasalização da vogal ou do ditongo anterior (CEGALLA, 1997, p. 29).
/l/	Deve-se utilizar o fonema /l/ somente quando a letra ***l*** estiver em início de sílaba não acompanhado da consoante ***h***, ou antecedido das consoantes ***f***, ***c*** ou ***p***. Quando apresentar-se no final de sílaba, deve-se usar o fonema /u/. Segundo Cegalla (1997, p. 29), "isto se deve à pronúncia normal brasileira, em que o *l* em final de sílaba, é vocalizado aproximadamente como um *u*: *alto* (aultu), *mal* (mául)".
/r/ /rr/ /r2/	A consoante ***r*** quando se apresenta geminada, possui uma pronúncia diferenciada, como, por exemplo, nas palavras *ca**rr**o* e *ca**r**o*. Isto porque a pronúncia do ***r*** geminado é formada na zona posterior da cavidade bucal, enquanto que o ***r*** singularmente é formado na zona anterior. Esta característica demanda a necessidade de dois fonemas /rr/ e /r/, respectivamente. O fonema /r/ será usado quando a consoante for encontrada no início ou meio da palavra e estiver seguido de vogal. Já o fonema /rr/ será usado quando encontrado o referido encontro consonantal. Devido a regras do sintetizador MBROLA, o fonema /r2/ substitui o fonema /r/ quando a letra ***r*** aparecer ao final de uma palavra ou anteceder uma consoante. Exemplos do uso do fonema /r2/ são *maio**r**, pa**r**te, ja**r**dim, ma**r***.
/a/ /@/ /am/	Os fonemas /a/, /@/ e /am/ representam respectivamente: a vogal oral ***a***, o dígrafo ***an*** (vogal nasal) e o dígrafo ***am*** (vogal nasal). Para no caso da letra ***a*** aparecer com o acento gráfico til, será usado o fonema /am/, pois o sintetizador apresentou uma melhor pronúncia de acordo com alguns testes.
/e/ /ee/ /em/	Para os dígrafos ***en*** e ***em***, existe apenas um fonema disponível no banco de sons. Isto se deve a similaridade da vocalização das vogais nasais, como, por exemplo, nas palavras: ***em**presa*, ***en**cargo*, *v**en**da*, *t**em**po*. Para estes dígrafos será utilizado o fonema /em/ na transcrição. Já o fonema /ee/ deve ser utilizado quando a vogal ***e*** receber o sinal diacrítico de acento agudo, como nos exemplos: ***pé***, *v**éu***, *c**éu***. As demais ocorrências da vogal ***e*** serão transcritas com o fonema /e/.
/i/ /im/	A exemplo da vogal ***e***, a letra ***i*** também apresenta apenas o fonema /im/ para os dígrafos ***in*** e ***im*** que sinalizam nasalização. Para as demais ocorrências da letra, usar-se-á o fonema /i/.
/o/ /oo/ /om/	Os fonemas da vogal ***o*** apresentam uma regra muito semelhante da vogal ***e***. Quando a letra ***o*** for vogal nasal, a transcrição ocorrerá com o fonema /om/, como nas palavras: ***om**bro*, ***on**tem*, *t**on**to*, *c**om**boio*. O fonema /oo/ apresenta uma vocalização aberta e será utilizado quando a vogal ***o*** receber um acento agudo. Em todos os outros casos da letra ***o***, a transcrição ocorrerá com o fonema /o/.
/u/ /um/	A última das vogais, quando se encontrar nos dígrafos ***um*** e ***un***, será transcrita pelo fonema /um/. Ambos os dígrafos possuem vocalização semelhantes e por isso o banco de fonemas possui apenas um som para a vogal nasal. As demais ocorrências da letra ***u*** serão transcritas com o fonema /u/.
/y/ /w/	Os fonemas /y/ e /w/ são classificados como semivogais. Cegalla (1997, p. 24) explica que "semivogais são os fonemas /i/ e /u/ átonos que se unem a uma vogal, formando com esta uma só sílaba". De acordo com isto, pode-se afirmar que o fonema /y/ aparece na transcrição quando o fonema /i/ antecede uma vogal ou quando uma vogal é acompanhada do fonema /i/. A mesma regra se aplica para o fonema /w/ sob a condição que o fonema /u/ esteja acompanhado de uma vogal ou uma vogal anteceda o fonema /u/.

Quadro 33 – Regras de transcrição dos fonemas

APÊNDICE B – Algoritmo de separação silábica

A seguir é apresentado o código completo do algoritmo de separação silábica (quadro 34).

```
procedure TSeparadorSilabico.SeparacaoSilabica(prPalavra : String);
var
  posletra  : Integer;
  ini       : Integer;
  final     : Integer;

  sPalavra  : String;
  posProximaVogal : Integer;

  conc : boolean; //concluido
  aeo  : boolean;

  function procuraProximaVogal(prPosicao : Integer) : Integer;
  begin
    Result := 0;
    while (prPosicao <= Length(sPalavra)) do
    begin
      if (pos(sPalavra[prPosicao],VOGAIS) > 0) then
      begin
        Result := prPosicao;
        exit;
      end;
      Inc(prPosicao);
    end;
  end;

begin
  sPalavra := LowerCase(Trim(prPalavra));
  posletra := 1;
  ini      := 1;
  final    := 0;
  while final < Length(sPalavra) do
  begin
    conc := false;
    aeo  := false;
    while not conc do
    begin
      if (posletra+1) > Length(sPalavra) then
      begin
        final := posletra;
        conc  := true;
      end
      else begin
//Procurar a próxima vogal da palavra e partir de posletra e atribuit a sua posição
a posletra
        if (posletra+1) > Length(sPalavra) then
        begin
          final := Length(sPalavra);
          conc  := true;
        end
        else begin
```

```
          if pos(sPalavra[posletra+1],VOGAIS) > 0 then //posletra + 1 ESTA EM VOGAL
          begin                          //iu
            if (pos(sPalavra[posletra+1],VOGAL_i+Vogal_u) > 0) and
               (pos(sPalavra[posletra+3],VOGAIS) <= 0) and
               ( ( (pos(sPalavra[posletra+2],'bdfkptv') > 0) and
(pos(sPalavra[posletra+3],'hlr') <= 0) ) or
                 ( (sPalavra[posletra+2] = 'c') and
(pos(sPalavra[posletra+3],'hlrk') <= 0) ) or
                 (sPalavra[posletra+2] = 'h') or
                 ((pos(sPalavra[posletra+2],'jlmrxz') > 0) and
(sPalavra[posletra+3] <> 'h') and (posletra+3 < Length(sPalavra)) ) or
                 ((sPalavra[posletra+2] = 'g') and (pos(sPalavra[posletra+3],'lr')
<= 0)) or (sPalavra[posletra+2] = 'q')
                ) then
            begin
              //Não permite que fiquem consoantes sozinhas
              if (pos(sPalavra[posletra],VOGAIS) > 0) and (final + 1 = posletra)
then
              begin
                final := posletra;
                conc  := true;
              end;
            end;

            //Tratar hiato separável "ie" "ue" (ex: am-bi-en-te, bu-ei-ro)
            if (pos(sPalavra[posletra+1],VOGAL_i+Vogal_u) > 0) and
               (pos(sPalavra[posletra+2],VOGAL_e) > 0) and
               (pos(sPalavra[posletra],'qg') <= 0) then
            begin
              final := posletra + 1;
              posletra := posletra + 1;
              conc  := true;
            end;

            if (conc = false) then
            begin
              if pos(sPalavra[posletra],'aeo') > 0 then
              begin
                if (pos(sPalavra[posletra+1],'aeoíú') > 0) then
                begin
                  final := posletra;
                  conc  := true;
                end
                else begin
                  aeo   := true;
                end;
              end
              else begin  //posletra está em vogal acentuada, ou em encontro
vocálico ao final de palavra
                if (pos(sPalavra[posletra],'àáéíóúâêîôûãõü') > 0) or
                   (pos(sPalavra[posletra+1],'àáéíóúâêîôûãõü') > 0) or
                   ( ((posletra+1) = Length(sPalavra)) and
(pos(sPalavra[posletra+1],'aeo') > 0)) or
                    (aeo = true) or (posletra = posletra + 1) then
                begin
                  //caso especial o i acentuado ("í"), que antecede "s" ou "n",
exceto quando "NH"
                  if ((sPalavra[posletra+1] = 'í') and (sPalavra[posletra+2] =
's')) or
                     ((sPalavra[posletra+1] = 'í') and (sPalavra[posletra+2] = 'n')
and  (sPalavra[posletra+3] <> 'h')) then
                  begin
                    posletra := posletra + 1;
                  end
                  else begin
                       //Regra para palavras com vogal com til: Ex: pão, joão, mãe
                    if ((pos(sPalavra[posletra+1],'ãõ') > 0) and
(pos(sPalavra[posletra+2],VOGAIS) > 0)) or
```

```
                      //Regra para palavras com vogal acentuada seguida de M ou N:
Ex: vigilÂNcia, ciÊNcia, retÉM, tambÉM
                      ((pos(sPalavra[posletra+1],'aeéâêî') > 0) and
(pos(sPalavra[posletra+2],'nm') > 0))  then
                    begin
                      //Caso tenha plural, deve-se adicionar o S a sílaba.
                      if (pos(sPalavra[posletra+3],'s') > 0) then
                      begin
                        final := posletra + 3;
                        posletra := posletra + 3;
                        conc  := true;
                      end
                      else begin
                        //Senão, finaliza a sílaba no M ou N
                        final := posletra + 2;
                        posletra := posletra + 2;
                        conc  := true;
                      end;
                    end
                    else begin
                      final := posletra + 1;
                      posletra := posletra + 1;
                      conc  := true;
                    end;
                  end;
                end;
              end;
            end;
            posletra := posletra + 1;
          end
          else begin //posletra + 1  ESTA EM CONSOANTE
            //A consoante inicial não seguida de vogal permanece na sílaba que a
segue
            if (posletra = 1) and (pos(sPalavra[posletra],VOGAIS) <= 0) and
(pos(sPalavra[posletra+1],VOGAIS) <= 0) then
            begin
              posletra := posletra + 2;
            end;

            While ((posletra + 2) <= Length(sPalavra)) and
(pos(sPalavra[posletra+2],VOGAIS) <= 0) and
                  ( ((pos(sPalavra[posletra+1],'bdfkptv') > 0) and
(pos(sPalavra[posletra+2],'hlr') <= 0)) or
                    ((sPalavra[posletra+1] = 'c') and
(pos(sPalavra[posletra+2],'hlrk') <= 0)) or
                    (sPalavra[posletra+1] = 'h') or
                    ((pos(sPalavra[posletra+1],'jlmnrsxz') > 0) and
(sPalavra[posletra+2] <> 'h')) or
                    ((sPalavra[posletra+1] = 'g') and
(pos(sPalavra[posletra+2],'lr') <= 0)) or
                    (sPalavra[posletra+1] = 'q')
                  ) do
            begin
              posletra := posletra + 1;
            end; //fim while

            //tratamento do encontro consonantal LH e NH
            if (pos(sPalavra[posletra],'ln') > 0) and (sPalavra[posletra+1] = 'h')
then
            begin
              posletra := posletra + 2;
              //Se LH ou NH estiverem seguidos se N ou M, que por sua vez seguidos
de outra consoante, deve-se adicionar o N ou M à silaba
              if (posLetra+3 <= Length(sPalavra)) and
                 (pos(sPalavra[posletra+1],'nm') > 0) and
                 (pos(sPalavra[posletra+2],CONSOANTES) > 0) then
              begin
                posletra := posletra + 1;
```

```
              end;
            end;

            //verificar se é um encontro consonantal inseparável
            //bl, br, cl, cr, dr, fl, fr, gl, gr, pl, pr, tl, tr, vr
            if (pos(sPalavra[posletra],'bcdfgptv') > 0) and
               (pos(sPalavra[posletra+1],'lrh') > 0) and
               (pos(sPalavra[posletra+2],VOGAIS) > 0) then
            begin
              posletra := posletra + 2;
              //Adiciona mais uma posição a sílaba caso o encontro seja acompanhado
de N ou M, não seguido de vogal, nem da consoante H. Ex:  queBRANdo, lemBRANdo,
              if (pos(sPalavra[posletra+1],'mn') > 0) and
                 (pos(sPalavra[posletra+2],VOGAIS) <= 0) and
                 (sPalavra[posletra+2] <> 'h') then
              begin
                posletra := posletra + 1;
              end;
            end;

            if ((posletra+2) > Length(sPalavra)) then
            begin
              final := posletra + 1;
            end
            else begin
              //procura a proxima vogal da palavra a partir de pos+2
              posProximaVogal := procuraProximaVogal(posletra+2);
              if (posProximaVogal <= 0) then
              begin
                final := Length(sPalavra);
                posletra := Length(sPalavra) + 1;
              end
              else begin
                final := posLetra;
                posletra := posletra + 1;
              end;
            end;
            conc := true;
          end;
        end
      end;
    end;

    FSilabasSeparador.Add(Copy(prPalavra,ini,(final-ini)+1));
    ini := posletra;
  end;

  FSilabasQtdLetras.Clear;
  for ini := 0 to FSilabasSeparador.Count - 1 do
  begin
    FSilabasQtdLetras.Add(IntToStr(Length(FSilabasSeparador[ini])));
  end;
end;
```

Quadro 34 – Algoritmo de separação silábica

APÊNDICE C – Algoritmo de tradução de números para sua forma por extenso

A seguir é apresentado o algoritmo completo para tradução de números para sua forma biunívoca cardinal (quadro 35).

```
const
  Unidades: array[1..19] of TNumeroStr = ('um', 'dois', 'três', 'quatro','cinco',
'seis', 'sete', 'oito', 'nove', 'dez', 'onze', 'doze',
  'treze', 'quatorze', 'quinze', 'dezesseis', 'dezessete', 'dezoito','dezenove');

  Dezenas: array[1..9] of TNumeroStr = ('dez', 'vinte', 'trinta',
'quarenta','cinqüenta', 'sessenta', 'setenta', 'oitenta', 'noventa');

  Centenas: array[1..9] of TNumeroStr = ('cem', 'duzentos',
'trezentos','quatrocentos', 'quinhentos', 'seiscentos', 'setecentos',
'oitocentos','novecentos');

  ErrorString = 'Valor fora da faixa';
  Min = 0.01;
  Max = 4294967295.4294967295;

implementation

function TNumeroPorExtenso.NumeroParaExtenso(prNumero: String): string;
var
  vValorInteiro  : Integer;
  vFrac          : String;
  vValorFrac     : Integer;
begin
  prNumero := StringReplace(prNumero,'.',',',[rfReplaceAll,rfIgnoreCase]);
  if (StrToFloat(prNumero) >= Min) and (StrToFloat(prNumero) <= Max) then
  begin
    //Verifica se tem fracao
    if (pos(',',prNumero) > 0) then
    begin
      //Traduz a parte inteira
      vValorInteiro := StrToInt64(Copy(prNumero,1,pos(',',prNumero)-1));
      Result := ConversaoRecursiva(vValorInteiro);
      //Traduz a parte fracionária
      Result := Result + ' vírgula';
      //Traduz os zeros da parte fracionaria
      vFrac := Copy(prNumero,pos(',',prNumero)+1,Length(prNumero));
      while vFrac[1] = '0' do
      begin
        Result := Result + ' zero';
        vFrac  := Copy(vFrac,2,Length(vFrac));
      end;
      //Traduz os números da parte fracionária
      if StrToInt(vFrac) > 0 then
      begin
        vValorFrac := StrToInt64(vFrac);
        Result := Result + ConversaoRecursiva(vValorFrac);
      end;
    end
    else begin
      Result := ConversaoRecursiva(StrToInt64(prNumero));
    end;
  end
  else
    raise ERangeError.CreateFmt('%g ' + ErrorString + ' %g..%g',[prNumero, Min,
Max]);
end;
```

```
function TNumeroPorExtenso.ConversaoRecursiva(N: LongWord): string;
begin
  case N of
    1..19:
            Result := Unidades[N];
    20, 30, 40, 50, 60, 70, 80, 90:
            Result := Dezenas[N div 10] + ' ';
    21..29, 31..39, 41..49, 51..59, 61..69, 71..79, 81..89, 91..99:
            Result := Dezenas[N div 10] + ' e ' + ConversaoRecursiva(N mod 10);
    100, 200, 300, 400, 500, 600, 700, 800, 900:
            Result := Centenas[N div 100] + ' ';
    101..199:
            Result := 'cento e ' + ConversaoRecursiva(N mod 100);
    201..299, 301..399, 401..499, 501..599, 601..699, 701..799, 801..899, 901..999:
            Result := Centenas[N div 100] + ' e ' + ConversaoRecursiva(N mod 100);
    1000..999999:
            Result := ConversaoRecursiva(N div 1000) + ' mil ' +
ConversaoRecursiva(N mod 1000);
    1000000..1999999:
            Result := ConversaoRecursiva(N div 1000000) + ' milhão '+
ConversaoRecursiva(N mod 1000000);
    2000000..999999999:
            Result := ConversaoRecursiva(N div 1000000) + ' milhões '+
ConversaoRecursiva(N mod 1000000);
    1000000000..1999999999:
            Result := ConversaoRecursiva(N div 1000000000) + ' bilhão ' +
ConversaoRecursiva(N mod 1000000000);
    2000000000..4294967295:
            Result := ConversaoRecursiva(N div 1000000000) + ' bilhões ' +
ConversaoRecursiva(N mod 1000000000);
  end;
end;
```

Quadro 35 – Algoritmo de tradução de números por extenso