Desenvolvimento de Software Livre Usando Reconhecimento e Síntese de Voz: O Estado da Arte para o Português Brasileiro

Nelson Sampaio Neto[1], **Erick Sousa**[1], **Valquíria Macedo**[1], **André G. Adami**[2] **e Aldebaro Klautau**[1]

[1]Laboratório de Processamento de Sinais – LaPS
DEEC, Universidade Federal do Pará – UFPA
Rua Augusto Correa, 1 – 660750-110 – Belém, PA, Brasil
http://www.laps.ufpa.br

[2]OGI School of Science & Engineering, OHSU
Portland, Oregon, EUA.

{nsampaio,erick,valquiria,aldebaro}@deec.ufpa.br,adami@bme.ogi.edu

Abstract. *Speech is a natural interface for human-machine communication. Speech (or voice) technology is a well-developed field when one considers the international community. There is a wide variety of academic and industrial software. The majority of them assumes a recognizer or synthesizer is available, and can be programmed through an API. In contrast, there are no such resources in public domain for Brazilian Portuguese. This work discusses some of efforts towards filling this gap and the state-of-art of speech-enabled applications.*

Resumo. *A fala é uma interface natural para a comunicação homem-máquina. A tecnologia de processamento de fala (ou voz) encontra-se bastante avançada e, em escala mundial, existe vasta disponibilidade de software tanto comercial quanto acadêmico. A maioria dos mesmos assume a disponibilidade de um reconhecedor e/ou sintetizador, o qual podem ser programados via uma API. Ao contrário do que acontece, por exemplo, para a língua inglesa, não existem atualmente reconhecedores ou sintetizadores de domínio público para o português brasileiro. Esse trabalho discute alguns esforços realizados nesse sentido e o estado da arte em aplicativos baseados em voz.*

1. Introdução

A tecnologia de voz possui grande potencial para a criação de aplicativos que possibilitem uma eficaz interação humano-computador, especialmente no atual contexto de acelerada miniaturização dos sistemas embarcados. Dentre os ramos dessa tecnologia, o reconhecimento e a síntese de voz são os abordados no presente trabalho. Resumidamente, dado um texto (em ASCII, por exemplo), a *síntese de voz* é o processo de geração de amostras de um sinal digital que deveria soar como se um humano tivesse dito o texto. A síntese de voz (ou TTS, de *text-to-speech*) não deve ser confundida com a simples digitalização, armazenagem e posterior *playback* de voz. O *reconhecimento de voz* (ou ASR, de *automatic speech recognition*) pode ser visto como o processo inverso, onde o sistema converte voz digitalizada em texto. Dependendo da aplicação, o resultado obtido pelo ASR pode

ser repassado para módulos como o de gerenciamento de diálogos, tutores inteligentes, processamento de linguagem natural (PLN) e outros.

A tecnologia de processamento de fala (ou voz) encontra-se bastante avançada e, em escala mundial, existe vasta disponibilidade de *software* tanto comercial quanto acadêmico. A maioria dos mesmos assume a disponibilidade de um reconhecedor e/ou sintetizador, conhecidos genericamente na literatura como *engines*, os quais podem ser programados via uma API (*application programming interface*). Ao contrário do que acontece para a língua inglesa, por exemplo, não existem atualmente reconhecedores ou sintetizadores de domínio público para o português brasileiro (PT_BR). Essa situação obviamente retarda a disseminação de uma tecnologia dentre os desenvolvedores brasileiros a qual, segundo estudo feito em 2003 pela *Allied Business Intelligence*, corresponde a um mercado de U$677 milhões em 2002, e projetado para U$5,3 bilhões em 2008. Aplicações de interesse incluem, por exemplo, auxílio a portadores de necessidades especiais [DOSVOX, 2005] e telefonia computadorizada (*call-centers* e similares).

No Brasil, grupos como o LPDS (UNICAMP) e LINSE (UFSC) detém amplo *know-how* no desenvolvimento de sistemas TTS e ASR.[1] Outro exemplo é o Instituto Genius, Manaus, o qual disponibiliza tecnologia ASR para aplicações como encaminhamento de chamadas telefônicas (*auto-attendant*) e outras que usem gramática do tipo CFG, a serem discutidas adiante, tais como as baseadas em reconhecimento de palavras isoladas para automação doméstica. Contudo, principalmente por falta de financiamento, não vigoram atualmente esforços para a criação de *engines* para o PT_BR visando sua disponibilização gratuita para a comunidade acadêmica. Um parágrafo extraído de [Rodrigues et al., 2004], publicado no SBC de 2004, ilustra claramente a situação dos pesquisadores nacionais, que são obrigados a usar *engines* para a língua inglesa: "*Outros pontos interessantes de serem abordados em trabalhos futuros são: acrescentar ao sistema novos idiomas para a síntese da voz, principalmente o português; e a alternativa de operar com vozes capturadas. Esse último ponto exigirá, no entanto, a integração com ferramentas de reconhecimento de voz...*".

O presente trabalho encontra-se organizado da seguinte maneira. Na Seção 2, é apresentada uma breve revisão das tecnologias TTS e ASR, identificando os esforços para o suporte ao PT_BR. Na Seção 3 aborda-se a construção de um aplicativo CALL (*computer-assisted language learning*) para o ensino da língua inglesa, no qual é usado ASR em inglês (baseado em SAPI) e TTS em português, sendo a mesma seguida pelas conclusões.

2. Estado da Arte em ASR e TTS

Nessa seção discute-se ASR e posteriormente TTS. No estado da arte em ASR, o complexo processo de decodificação do sinal realizado pela mente humana é imitado pelo computador com base em modelos probabilísticos. Atualmente, a grande maioria dos sistemas para voz contínua e grandes vocabulários (LVCSR, de *large-vocabulary continuous speech recognition*) é baseada em cadeias escondidas de Markov (HMMs, de *hidden Markov models*) [Huang et al., 2001]. Esses sistemas usam um *front end* para converter

[1] O desenvolvimento desses trabalhos pode ser acompanhado pelas publicações no workshop TIL do SBC e no Simpósio Brasileiro de Telecomunicações (SBT), os quais são atualmente os principais veículos de divulgação da ciência nacional nessa área.

o sinal de voz digitalizado em uma matriz $\mathbf{X}$ de parâmetros, e buscam a seqüência de palavras W que maximiza a probabilidade condicional

$$\hat{W} = \arg\max_{W} P(W|\mathbf{X}).$$

Na prática, usa-se a regra de Bayes para implementar a busca através de:

$$\hat{W} = \arg\max_{W} \frac{P(\mathbf{X}|W)P(W)}{P(\mathbf{X})} = \arg\max_{W} P(\mathbf{X}|W)P(W)$$

com $P(\mathbf{X})$ sendo desprezado pois não depende de W. Para cada W, os valores de $P(\mathbf{X}|W)$ e $P(W)$ são fornecidos pelos modelos *acústico* e de *linguagem* (ou língua [Pessoa et al., 1999]), respectivamente.

O modelo de linguagem precisa fornecer a probabilidade conjunta $P(W) = P(w_1, w_2, ..., w_n)$ da seqüência de palavras $W = w_1, w_2, ..., w_n$, a qual pode ser decomposta em:

$$P(W) = P(w_1)P(w_1|w_2)P(w_3|w_1, w_2)...P(w_n|w_1, w_2, ..., w_{n-1}).$$

Na prática, é inviável estimar robustamente as probabilidades $P(w_i|w_1, ...w_{i-1})$ mesmo para valores moderados de i. A solução típica, chamada modelo N*-grama* é assumir que $P(w_i|w_1, ...w_{i-1})$ depende apenas das $N-1$ prévias palavras. Caso $N = 3$ tem-se um trigrama $P(w_i|w_{i-2}, w_{i-1})$, enquanto $N = 2$ e 1 são chamados bigrama $P(w_i|w_{i-1})$ e unigrama $P(w_i)$, respectivamente. Em um LVCSR com vocabulário em torno de 60 mil palavras, é comum o modelo de linguagem ser composto por algo em torno de 40 milhões de trigramas e 20 milhões de bigramas (e 60 mil unigramas).

Em contraste com os modelos N-grama, alguns sistemas ASR utilizam um modelo de linguagem não-probabilístico, baseado em gramáticas livres de contexto (CFG, de *context-free grammars*). As CFG são mais apropriadas para aplicações similares às de comando-e-controle, enquanto as N-grama são úteis em aplicações de ditado (daí serem chamadas *dictation grammars*), criação automática de legendas (*captioning*), e outras onde a *perplexidade* [Huang et al., 2001] seja relativamente alta.

Além do modelo de linguagem, o léxico é um outro recurso importante. O ASR exige que cada palavra do vocabulário tenha sua correspondente transcrição fonética. Se a inclusão de uma nova palavra é feita *off-line*, um especialista pode mapeá-la nos *alofones* [Huang et al., 2001] correspondentes. O processo *on-line*, onde o usuário acrescenta uma palavra a partir apenas de sua ortografia, exige um procedimento relativamente sofisticado conhecido como *letter-to-sound* ou *grapheme-to-phoneme* [Black et al., 1998] (o algoritmo responsável é às vezes chamado *phoneticizer* [PSTL, 2004]).

Outros recursos presentes em ASR são o modelo acústico e o *front end*. Do ponto de vista do programador, os mesmos são menos visíveis, e por isso não serão aqui detalhados.

Em termos de TTS, o principal desafio é fazer com que a voz sintetizada soe menos robotizada e mais humana. Dentre os fatores que dificultam atingir esse objetivo, encontra-se a modelagem da prosódia, uma das principais responsáveis para que a voz sintetizada carregue o ritmo e emoção que um ser humano empregaria. O melhor exemplo

da evolução da tecnologia TTS é o sistema da AT&T [AT&T, 2005], o qual atinge uma voz bastante próxima da natural.

Os sistemas TTS evoluíram de um paradigma *knowledge-based*, onde lingüistas e engenheiros trabalhavam muito mais afinados, para uma pragmática abordagem *data-driven*. Nessa direção, o *back end* do TTS, às vezes chamado de sintetizador, por ser o módulo responsável pela geração das amostras do sinal digital, evoluiu da técnica conhecida como síntese por *formantes* [Allen et al., 1987] para a *concatenativa* [Dutoit, 2001], onde segmentos de voz (representando, por exemplo, *difones*) são armazenados para posterior concatenação. Uma das técnicas em voga é a utilização de HMMs para o treinamento automático de sistemas TTS [HTS, 2005]. Além dos *engines*, uma API é essencial para o desenvolvimento de aplicativos baseados em voz.

Empresas como a IBM e Microsoft possuem soluções que adotam padrões como VoiceXML e Salt, além de promoverem o uso de APIs. Esse trabalho discute duas delas, as quais bem representam as opções disponíveis. A primeira é a SAPI (*speech API*) da Microsoft [SAPI, 2005] e a segunda a JSAPI (*Java speech API*) da Sun [JSAPI, 2005]. A SAPI, como qualquer produto Microsoft, foi projetada para o sistema Windows, enquanto a JSAPI é a que melhor se encaixa em soluções que exijam portabilidade.

Ambas permitem o emprego das tecnologias de TTS e ASR. A JSAPI deixa a desejar num ponto de extrema importância em TTS, ao não permitir o redirecionamento das amostras de áudio para arquivos e/ou dispositivos além da saída de áudio padrão. Porém, ambas suportam o uso de gramáticas CFG e de ditado (probabilísticas). A SAPI utiliza-se da XML (*Extensible Markup Language*) para incluir "marcas" prosódicas (formatar) o texto a ser sintetizado, ou seja, configurar os parâmetros de síntese de voz como velocidade, afinação, timbre e língua. Já com JSAPI, também é possível trabalhar com esses parâmetros tanto através de invocação de procedimentos e/ou métodos, como pela JSML (*Java Synthesis Markup Language*).

A escolha de qual interface utilizar depende de vários aspectos, que vão desde critérios técnicos, como o sistema operacional, a critérios bastante subjetivos, como facilidade de uso. A SAPI utiliza COM (*Component Object Model*) e por isso pode ser utilizada a partir de qualquer linguagem que suporte COM. A JSAPI não é uma interface do estilo COM, logo a linguagem empregada no desenvolvimento do sistema deve ser a mesma utilizada para escrever a JSAPI, no caso Java. A JSAPI sugere que o aplicativo se comunique com o engine a partir de *Java Events*. Assim, a JSAPI pode ser utilizada em qualquer sistema operacional para o qual exista uma máquina virtual Java. Em contraste, a SAPI é restrita a sistemas operacionais Windows.

No que diz respeito a *engines*, existe uma gama de *softwares* que suportam SAPI e/ou JSAPI. Esse é um ponto importante, pois permite ao desenvolvedor avaliar a qualidade de *engines* de fabricantes diferentes e escolher o que melhor se aplica às funcionalidades do seu sistema. A SAPI é adotada pela maioria das empresas do setor, tais como Dragon, IBM, e Lucent. A JSAPI é bem menos difundida, e uma das exceções é o suporte à parte ASR dessa interface no produto ViaVoice da IBM.

3. Exemplo de Aplicação: "CALL"

Nessa seção discute-se brevemente um *software* para aprendizado de língua estrangeira com o auxílio de computador. O mesmo apresenta palavras e frases para o aluno, ou

mesmo atividades de ditado. O *software* utiliza agentes animados para tornar mais natural a interface com os usuários, podendo-se não apenas mostrar texto na tela do computador, mas também permitir que o aluno ouça a pronúncia correta de uma palavra ou frase, através do uso de síntese de voz por parte dos agentes. O protótipo de CALL que está sendo desenvolvido aborda o ensino da língua inglesa, em função da maior disponibilidade de *engines*. Em [Silva et al., 2004] são descritas as atividades de desenvolvimento de ASR para PT_BR.

Utilizando-se os *engines* disponibilizados pela Microsoft para ASR e TTS, pode-se interagir com o usuário através da língua inglesa. Contudo, levando-se em conta que o usuário é muitas vezes uma criança, é importante se poder usar TTS em língua portuguesa para instruções e *feedback*. Essa situação é compartilhada com outros projetos em desenvolvimento no Brasil (e.g., [Rodrigues et al., 2004]). Para o presente trabalho, utiliza-se o *software* do CSLU [CSLU, 2005], o qual incorpora suporte ao PT_BR através da "voz" AGA.

Apesar de relativamente precário, o suporte à TTS em PT_BR do CSLU é uma das contribuições de um dos autores às pesquisas na área. Em resumo, o processo de geração da voz AGA foi baseado na síntese por difones, um dos métodos mais populares para criação de voz sintética [Lenzo and Black, 2000]. O difone é a região entre os "centros" de dois *fones* adjacentes. O centro de uma realização fonética é a região mais estável da mesma, enquanto que a transição de um *fone* para outro contém mais variação (efeitos da co-articulação), o que torna difícil a modelagem. Após a seleção do conjunto de *fones*, os difones são coletados utilizando palavras que possuam o respectivo difone no meio [Isard and Miller, 1986]. A coleta foi feita em uma câmara anecóica, com equipamento de gravação de alta-qualidade e um laringógrafo. O laringógrafo é utilizado para garantir a qualidade da gravação e extrair informação sobre o *pitch* e pulso glotal. Após a coleta, as palavras foram alinhadas temporalmente com os difones e segmentadas posteriormente. Finalmente, os modelos prosódico e léxico são gerados para produzir uma voz o mais natural possível e lexicamente correta.

O *toolkit* do CSLU é uma plataforma para o desenvolvimento de aplicativos de diálogos. O *software* provê quatro níveis de interface para o desenvolvimento de aplicativos. No nível mais baixo, os desenvolvedores podem usar código C para acessar todos os componentes, tais como o *engine* ASR. Não há, contudo, suporte para uma API genérica como a SAPI ou JSAPI. Apesar de não ser a solução ideal, a seguir discute-se a estratégia utilizada para sintetizar em PT_BR usando CSLU. Uma das diretrizes adotadas, foi buscar uma solução simples, a qual não envolvesse modificações no código do CSLU. Assim, optou-se pela utilização de arquivos *batch* que invocavam o TTS do CSLU, sob o comando do aplicativo. Os passos adotados foram os seguintes ($cslu é a pasta onde o *software* CSLU foi instalado).

Dentro do diretório $cslu/Festival/1.4.2/festival/lib, o aplicativo cria um arquivo com o texto que deseja sintetizar. Nesse exemplo, assume-se que o nome desse arquivo é "comandos.txt". O comandos.txt possui o seguinte conteúdo:

```
(voice_aga_diphone) (SayText "ENTRE O TEXTO AQUI")
```

Então, no diretório $cslu/Toolkit/2.0/bin, criou-se uma cópia do arquivo Festival.bat (chamada aqui de "call.bat"), e modificou-se sua última linha para:

```
$CSLU\Festival\1.4.2\bin\festival.exe --batch "comandos.txt"
```

Após esses passos, basta ao aplicativo criar um arquivo comandos.txt e invocar o arquivo call.bat. Por exemplo, a partir de um programa C, bastaria a chamada:

```
system("$cslu\\Toolkit\\2.0\\bin\\call.bat");
```

Nota-se que a solução apresentada acima é ineficiente no sentido de que o processo de escrita e leitura em disco é lento. Contudo, o mesmo é simples e pode ser usado em situações onde a velocidade e qualidade do TTS não são críticas.

4. Conclusões

Como atualmente são sistemas *data-driven*, o reconhecimento e a síntese de voz se beneficiam da disponibilidade de corpora com grande volume de dados. Existe uma quantidade razoável de textos para estudos de modelagem de linguagem para a língua inglesa, português europeu e outras (vide [LDC, 2005]). Todavia, há poucos recursos acessíveis quando se trata do PT_BR. Essa lacuna é ainda maior quando se trata de voz digitalizada para treinamento do modelo acústico e desenvolvimento de TTS. A inexistência de uma grande base de dados e *engines* não só atrasa as pesquisas, mas também impede que os resultados obtidos por diferentes grupos de pesquisa sejam comparados diretamente.

O presente trabalho discutiu o estado da arte em ASR e TTS, e também avaliou a utilização de APIs. Abordou-se a construção de um aplicativo para o ensino da língua inglesa, no qual é usado ASR em inglês (baseado em SAPI) e TTS em português (baseado no *toolkit* do CSLU), disponibilizado por um dos autores.

References

Allen, J., Hunnicutt, M. S., Klatt, D. H., Armstrong, R. C., and Pisoni, D. B. (1987). *From text to speech: The MITalk system*. Cambridge University Press.

AT&T (Visited in March, 2005). http://www.research.att.com/projects/tts/demo.html.

Black, A., Lenzo, K., and Pagel, V. (1998). Issues in building general letter to sound rules. In *ESCA Synthesis Workshop, Australia 1998*.

CSLU (Visited in March, 2005). http://cslu.cse.ogi.edu/toolkit/.

DOSVOX (Visited in March, 2005). http://intervox.nce.ufrj.br/dosvox/.

Dutoit, T. (2001). *An Introduction to Text-To-Speech Synthesis*. Kluwer.

HTS (Visited in March, 2005). http://hts.ics.nitech.ac.jp/.

Huang, X., Acero, A., and Hon, H.-W. (2001). *Spoken language processing*. Prentice-Hall.

Isard, S. and Miller, D. (1986). Diphone synthesis techniques. In *Proceedings of the IEE International Conference on Speech Input/Output*, pages 77–82.

JSAPI (Visited in March, 2005). http://java.sun.com/products/java-media/speech/.

LDC (Visited in March, 2005). http://www.ldc.upenn.edu.

Lenzo, K. A. and Black, A. W. (2000). Diphone collection and synthesis. In *ICSLP*.

Pessoa, L., Violaro, F., and Barbosa, P. (1999). Modelo de língua baseado em gramática gerativa aplicado ao reconhecimento de fala contínua. In *XVII Simpósio Brasileiro de Telecomunicações*, pages 455–458.

PSTL (2004). Patent EP984430 - speech recognizer with lexicon updateable by spelled word input, http://gauss.ffii.org/patentview/ep984430.

Rodrigues, P. L., Feijó, B., and Velho, L. (2004). Expressive talking heads: uma ferramenta de animação com fala e expressão facial sincronizadas para o desenvolvimento de aplicações interativas. In *Proceedings of Webmídia. SBC*.

SAPI (Visited in March, 2005). http://www.microsoft.com/speech/.

Silva, E., Pantoja, M., Celidônio, J., and Klautau, A. (2004). Modelos de linguagem n-grama para reconhecimento de voz com grande vocabulário. In *TIL2004 - 2º. Workshop em Tecnologia da Informação e da Linguagem Humana*.