Desenvolvimento e Comparação de Reconhecedores de Fala Embarcados e Distribuídos para Android

Cassio Batista, Thiago Coelho, Bruno Haick, Nelson Neto e Aldebaro Klautau

***Resumo*— Este trabalho compara dois sistemas de reconhecimento de fala que podem ser usados no desenvolvimento de aplicativos para Android: Julius em modo servidor e Google. Parte do suporte a Português Brasileiro para o Julius foi desenvolvido pelos autores no contexto do projeto FalaBrasil. O Julius também utilizou o servidor do FalaBrasil para prover reconhecimento distribuído via Internet, de maneira similar ao sistema da empresa Google. São apresentadas comparações entre os mesmos em termos de taxa de acerto (acurácia) e custo computacional.**

***Palavras-Chave*— Android, Português Brasileiro, reconhecimento de fala.**

***Abstract*— This paper compares two speech recognition systems that can be used to develop applications for Android: Julius on server mode and Google. Part of the support to Brazilian Portuguese for Julius was developed by the authors in the context of the FalaBrasil project. Julius also used the FalaBrasil server to provide distributed recognition via Internet, similar to the Google system. The systems are compared with respect to accuracy rate and computational cost.**

***Keywords*— Android, Brazilian Portuguese, speech recognition.**

I. Introdução

O presente trabalho apresenta um aplicativo desenvolvido sobre a plataforma Android 2.2 com suporte a reconhecimento de voz em Português Brasileiro através de duas ferramentas: Julius [1] e Google [2], ambos operando em modo distribuído. O reconhecedor PocketSphinx [3] em ambiente embarcado também foi testado, mas os resultados obtidos foram inconclusivos e serão melhor reportados em outro trabalho.

O Julius utiliza algumas ferramentas disponibilizadas pelo grupo de pesquisa FalaBrasil da Universidade Federal do Pará, como modelos acústicos [4] e dicionário fonético [5].

A contribuição do trabalho é fazer uma comparação entre as ferramentas de reconhecimento de fala em dispositivos móveis e indicar desenvolvimentos futuros.

II. Descrição do Aplicativo e Ambiente de Testes

Para que a comparação tenha escopo bem definido, foi utilizado um aplicativo específico chamado Rotas. O Rotas é um aplicativo que apresenta a rota dos ônibus da cidade de Belém utilizando ferramentas do Google Maps e banco de dados SQL (com SQLite). Quando o usuário pressiona o botão *start* e fala o local que deseja ir (rua, ponto turístico, etc), o sistema detecta onde a pessoa se encontra via GPS e mostra na tela do aparelho uma lista de ônibus que fazem o trajeto solicitado. Após escolher uma linha de ônibus, é exibido um mapa destacando a rota que o mesmo irá percorrer.

O dispositivo móvel utilizado para os testes foi o Samsung Galaxy Y GT-S6102B, com processador ARMv6 BCM21553 e 290 MB de RAM. É importante ressaltar que o Julius utiliza um botão adicional de *stop* para terminar a gravação da fala, o que não é necessário para o Google, que possui algoritmos que detectam o silêncio e interrompem automaticamente a gravação.

As duas ferramentas foram comparadas em termos do uso de CPU, consumo de RAM, tempo de retorno e acurácia. Os dados de consumo de RAM e uso de CPU foram obtidos através da ferramenta ADB 1.0.31 [2] com o seguinte teste: o botão de *start* foi clicado três vezes; a cada *click*, uma sentença era reproduzida ao microfone do aparelho e a ferramenta de reconhecimento era acionada.

O Google não dá suporte a arquivos. Assim, para verificar a viabilidade (acurácia e tempo) das ferramentas consideradas, 235 sentenças (wav) pré-gravadas com uma frequência de amostragem de 8 kHz foram reproduzidas para cada reconhecedor utilizando as mesmas caixas de som e o mesmo ambiente acústico controlado. Para o Julius, a transcrição de cada arquivo se configura como uma possível sentença de gramática livre de contexto, enquanto que a ferramenta do Google utiliza um modelo de linguagem próprio (da empresa).

Neste trabalho, o *average real time factor* ($\overline{xRT}$) foi definido como a média entre o tempo total do processo T_{ret} (desde o *click* no botão *start* até o retorno da sentença) dividido pela duração do áudio T_{aud} de cada arquivo:

$$\overline{xRT} = \frac{1}{N_s} \times \sum_{i=1}^{N_s} \frac{T_{ret_i}}{T_{aud_i}}, \tag{1}$$

onde o número de sentenças é $N_s = 235$. Essa métrica foi adotada por conta da impossibilidade de se obter alguns dados do Google, como o tempo de decodificação.

A acurácia foi calculada a partir da taxa de erro por sentença reconhecida (SER) e taxa de erro por palavra (WER) dada por:

$$WER = \frac{D + S + I}{N_p}, \tag{2}$$

onde N_p é o número de palavras na sentença de entrada, D, S e I são, respectivamente, os números de erros de deleção, substituição e inserção na sentença reconhecida.

Laboratório de Processamento de Sinais - LaPS, Faculdade de Engenharia da Computação, Universidade Federal do Pará, Belém - PA, Brasil, E-mails: {cassio.batista, thiago.coelho}@itec.ufpa.br, {bghaick, nelsonneto, aldebaro}@ufpa.br

III. Resultados e Conclusões

A Figura 1 mostra que não há muita diferença entre o consumo de RAM pelas duas ferramentas *online*. O Google consome em média pouco acima de 9,3 MB, enquanto o Julius, 10 MB. Ambos ficaram com números próximos aos do aplicativo em estado ocioso.

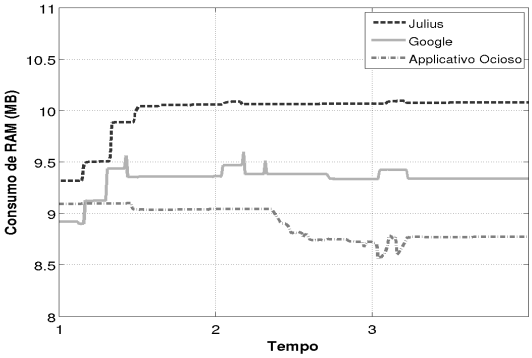

Fig. 1. Comparação de consumo de memória RAM. Os reconhecedores foram acionados três vezes, nos tempos 1, 2 e 3.

Da mesma forma, como ilustra a Figura 2, o uso de CPU pelo Google e Julius não ocupa muito a CPU, tendo uma média de consumo em torno de 20%.

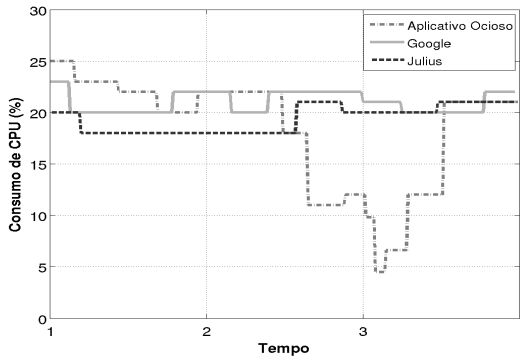

Fig. 2. Comparação de uso de CPU. Os reconhecedores foram acionados três vezes, nos tempos 1, 2 e 3.

A Tabela I mostra uma comparação entre os valores de $\overline{xRT}$ para as ferramentas. Os dois sistemas distribuídos foram testados com acesso via rede celular 3G e Wifi. O atraso da rede 3G se mostrou demasiado. Portanto, os resultados reportados via $\overline{xRT}$ são para uso de rede Wifi.

TABELA I
Comparação do tempo de retorno entre as ferramentas.

Ferramenta	T_{ret} (seg)	$\overline{xRT}$
Google	4,524	1,353
Julius	4,499	1,325

Google e Julius se mostraram bastante eficientes em termos de taxa de acerto. O primeiro, mesmo não utilizando gramáticas livres de contexto, obteve uma SER de apenas 7,66%. Já o Julius, utilizando uma gramática com 482 palavras e 235 sentenças possíveis, obteve 6,80%.

Com relação a WER, o Google obteve uma taxa melhor do que a do Julius, 3,13% contra 4,25%. Porém, alguns erros gramaticais (não fonéticos) foram observados em seus resultados. Por exemplo, a frase "bernardo sayão" foi escrita incorretamente como "bernardo saião". Já o Julius não apresentou esse tipo de erro por conta do uso de gramáticas personalizadas.

A Tabela II mostra uma comparação entre a taxa de erro por sentença e por palavra para os dois reconhecedores.

TABELA II
Comparação de acurácia entre as ferramentas.

Ferramenta	SER (%)	WER (%)
Google	7,66	3,13
Julius	6,80	4,25

Pode-se concluir que os sistemas avaliados obtiveram uma boa acurácia, mas o fator tempo de retorno deixou a desejar na rede 3G. O Julius se mostrou uma boa alternativa, pois permite impor gramáticas e é livre. Cogita-se que o Google venha cobrar de maneira mais agressiva pelo serviço no futuro, mas é a opção caso o aplicativo exija ditado, ou seja, não possa ser estruturado a partir de uma gramática. Como trabalho futuro, pretende-se realizar testes com envio de áudio via *streaming*, visando melhorar o desempenho com o Julius.

Testes experimentais com o PocketSphinx em ambiente embarcado também foram realizados. Para isso, foram utilizados modelos acústicos específicos para o Português Brasileiro construídos pelo grupo de pesquisa FalaBrasil [6]. Há cenários onde não existe acesso a Internet, tornando o reconhecimento em nuvem simplesmente inviável. Já a principal vantagem do PocketSphinx é não depender de rede de dados.

Sabe-se que por usar ponto fixo e várias simplificações, o PocketSphinx tem a WER maior do que a de um *software* equivalente concebido para *desktop* ou servidor. Ainda assim, constatou-se a necessidade de um maior esforço em desenvolvimento, dado que as taxas de erro obtidas preliminarmente com gramática controlada mostraram-se muito elevadas.

Referências

[1] Akinobu Lee, Tatsuya Kawahara, and Kiyoshiro Shikano, "Julius - an open source real-time large vocabulary recognition engine," *Proc. European Conference on Speech Communication and Technology*, pp. 1691–1694, 2001.

[2] "Android Developers," `http://developer.android.com/`.

[3] David Huggins-Daines, Mohit Kumar, Arthur Chan, Alan W Black, Mosur Ravishankar, and Alex I. Rudnicky, "Pocketsphinx: a free, real-time continuous speech recogntion system for hand-held devices," *Proceedings of ICASSP*, pp. 185–188, May 2006.

[4] Nelson Neto, Carlos Patrick, Aldebaro Klautau, and Isabel Trancoso, "Free tools and resources for Brazilian Portuguese speech recognition," *Journal of the Brazilian Computer Society*, vol. 17, pp. 53–68, 2011.

[5] Ana Siravenha, Nelson Neto, Valquíria Macedo, and Aldebaro Klautau, "Uso de regras fonológicas com determinação de vogal tônica para conversão grafema-fone em Português Brasileiro," *7th International Information and Telecommunication Technologies Symposium*, 2008.

[6] Rafael Oliveira, Pedro Batista, Nelson Neto, and Aldebaro Klautau, "Baseline acoustic models for Brazilian Portuguese using CMU sphinx tools," *12th International Conference on Computational Processing of the Portuguese Language*, pp. 375–380, 2012.