Identificação de Locutor
Utilizando MFC e GMMs

Ramon Duarte de Melo
André Ribeiro Queiroz
Universidade Federal do Rio de Janeiro
9 de julho de 2019

Resumo

Realizamos uma implementação de identificação de locutor utilizando a técnica de *Mel-frequency cepstrum* (MFC) para a extração das informações e modelos misturados de gaussianas (GMM) para os classificadores. Foram realizadas variações de duração da janela, afastamento entre janelas subsequentes (offset), quantidade de coeficientes de MFC e função de janelamento. O sistema foi implementado com base na biblioteca `python_speech_features` e nos módulos de métodos numéricos `scipy` e `numpy`. Observamos que o *offset* é o maior preditor de melhora da taxa de sucesso, produzindo resultados melhores quanto menor o afastamento (e maior a sobreposição). O número de coeficientes também foi significativo, com os melhores resultados sendo observados para as baterias de testes que envolviam 20 MFCCs. A duração da janela e a função de janelamento afetaram pouco as taxas de sucesso neste trabalho.

Palavras-chave: Mel-frequency cepstrum, modelos misturados de gaussianas, identificação de locutor

Introdução

A motivação inicial para este trabalho é a aplicação de identificação de locutor em cenários onde desempenho crítico não é exigido. Por exemplo, podemos imaginar um software de conferência que conecte usuários com níveis distintos de hierarquia ou privilégios. Quando um usuário com alto privilégio (por exemplo, o CEO de uma companhia) falar, o software poderá identificá-lo como de alto privilégio e proceder com o silenciamento dos microfones dos demais usuários, evitando interrupções ou conversas convolutas.

O sistema foi construído com a biblioteca `python_speech_features`, que fornece o método `mfcc()` para a extração dos coeficientes de *Mel-frequency cepstrum*, bem como de seus diferenciais (ou deltas, ou coeficientes dinâmicos). Em seguida, os classificadores foram construídos

com o método `scipy.gmm()`, que permite a gravação dos modelos sob arquivos `.gmm` que podem ser manipulados de forma ágil no momento de testes. Para o armazenamento persistente das informações, foram utilizados arquivos de texto plano, por não serem dados sensíveis e por serem fáceis de serem armazenados em repositórios de sistemas de versionamento como o `git`.

Os resultados foram analisados num `jupyter-notebook` através de tabelas de dados (*dataframes*) do `pandas` e métodos numéricos do `numpy`. Os gráficos foram produzidos pela biblioteca `matplotlib.pyplot`. Ao longo de todo o trabalho, a linguagem utilizada foi *Python 3.6.7*.

Implementação

O seguinte pipeline foi construído para o sistema:

1. Dados dos locutores alimentam o treinamento dos modelos.
2. Os sinais são amostrados e janelados durante o pré-processamento.
3. Os coeficientes e seus deltas são extraídos do dataset de treino.
4. Modelos misturados de gaussianas são utilizados para a modelagem estatística, de forma a tentar predizer a distribuição de MFCCs.
5. Os GMMs são gravados em disco para quando forem necessários ao teste.
6. Dados dos locutores para serem adivinhados alimentam o sistema pela interface de testes.
7. Os classificadores (GMMs treinados) são carregados do disco para produzirem probabilidades de cada locutor.
8. O sistema opta pelo locutor com maior probabilidade e retorna sua identificação.
9. Uma nota 1 é atribuída ao resultado caso o locutor retornado seja o mesmo do locutor testado; caso contrário, é atribuída a nota 0.

O dataset utilizado possui 34 locutores com 10 observações cada e foram adquiridos do fórum *VoxForge.com*. Os áudios foram disponibilizados sob a licença *Creative Commons* e, portanto, podem ser utilizados livremente, desde que a fonte seja mencionada.

Metade do dataset foi utilizado para o treinamento, especificamente as 5 observações mais longas de cada locutor. A maioria esmagadora dos áudios possuem entre 20 e 30 segundos. No entanto, há áudios que se aproximam dos 60 segundos.

Em todo caso, este dataset possui áudios incomumente longos, visto que a maioria dos sistemas de identificação de locutor trabalham com poucos segundos. A outra metade foi utilizada para os testes, tendo a maioria deles sido feitos com áudios de cerca de 10 segundos. Os áudios estão todos amostrados em 16 kHz, com 16 bits por amostra.

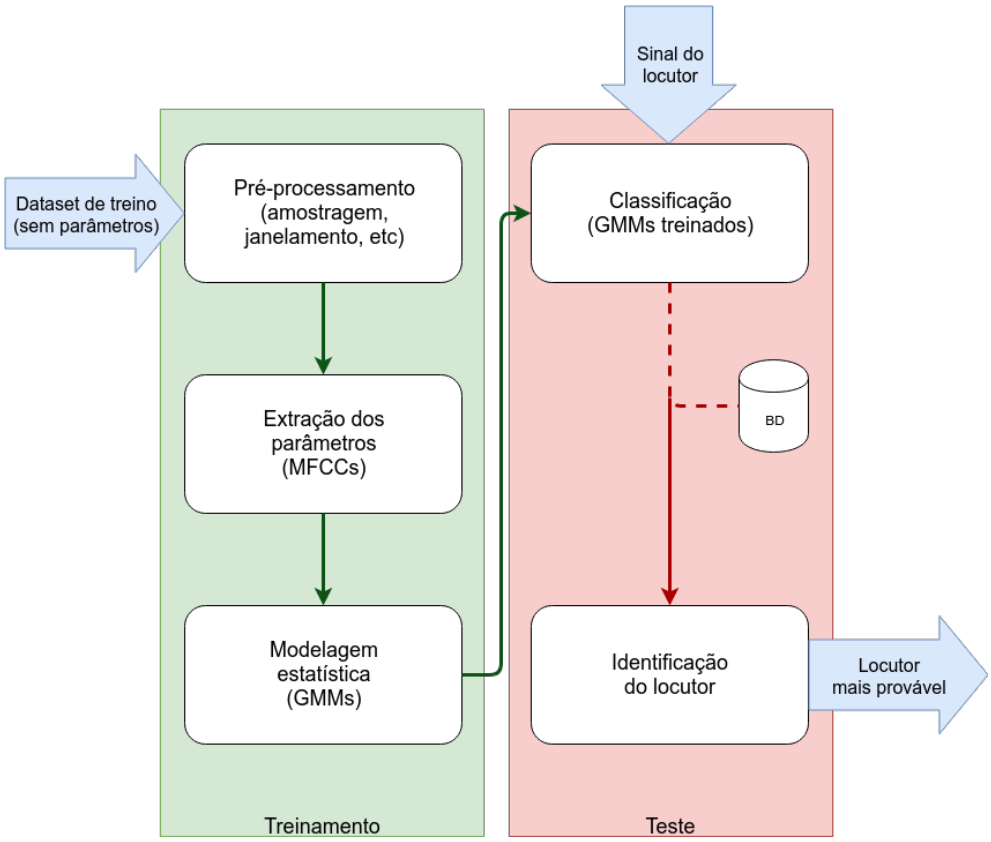

Figura 1: Arquitetura do sistema de identificação de locutor.

Os parâmetros variáveis no sistema são:

1. Número de coeficientes: qualquer número de MFCCs estáticos pode ser inserido como entrada. Entretanto, o número de coeficientes dinâmicos será obrigatoriamente igual ao de estáticos. Somente os primeiros diferenciais podem ser utilizados. Para este trabalho, foram utilizados $12 + 12$, $16 + 16$, $20 + 20$ e $24 + 24$ coeficientes.
2. Duração da janela: foram utilizados valores típicos para este parâmetro: 20, 25 e 30 ms.
3. Afastamento entre janelas (*offset*): o valor mais comumente utilizado na literatura é de 5 ms, mas também foram testados 10, 15 e 20 ms.
4. Função de janelamento: estes parâmetros são passados diretamente ao `scipy` (sem filtragem ou *parsing*) e, portanto, todas as funções de janelamento oferecidas por sua API estão disponíveis, como Blackman, Poisson, cossenoide, gaussiana, Hamming, Hanning e triangular. Para este trabalho, foram consideradas as funções de Blackman e de Hamming.

Os resultados obtidos foram analisados através da taxa de sucesso, do cálculo do desvio-padrão de suas distribuições, e de suas correlações lineares.

	mfccs	window	dist	fn	count
6	12	0.020	0.020	Blackman	0.823529
7	12	0.020	0.020	Hamming	0.823529
14	12	0.025	0.020	Blackman	0.847059
15	12	0.025	0.020	Hamming	0.847059
22	12	0.030	0.020	Blackman	0.847059
23	12	0.030	0.020	Hamming	0.847059
30	16	0.020	0.020	Blackman	0.852941
31	16	0.020	0.020	Hamming	0.852941
38	16	0.025	0.020	Blackman	0.882353
39	16	0.025	0.020	Hamming	0.888235
56	20	0.025	0.005	Blackman	1.000000
57	20	0.025	0.005	Hamming	1.000000
88	24	0.030	0.005	Blackman	1.000000
89	24	0.030	0.005	Hamming	1.000000

Figura 2: Piores (10 primeiras linhas) e melhores (4 últimas linhas) combinações de acordo com a taxa de acerto da bateria de testes.

Conclusão

Os resultados obtidos exibiram, no geral, taxas de sucesso bastante elevadas, com nenhuma das combinações produzindo taxas de sucesso inferiores a 80%. Quatro combinações acertaram todas as 170 anotações e receberam a taxa de sucesso de 100%. A maioria das demais oscilou entre 90 e 100% de acerto.

O parâmetro com maior impacto sobre a taxa de sucesso foi o afastamento entre uma janela e outra. Sua correlação com a taxa de acerto, em módulo, foi a maior deste trabalho, indicando que a redução das sobreposições afeta muito negativamente a eficácia do sistema.

O impacto do número de coeficientes foi o mais significativo depois do *offset*. Porém, nesta distribuição é possível observar que o número ótimo de coeficientes aproxima-se de $20 + 20$, começando a cair quando este valor é superado.

A função de janelamento teve quase nenhum impacto sobre a taxa de acerto. Observe que a correlação com a função de janelamento não é nula, mas desprezível, sendo da ordem de 10^{-15} ponto percentual. Ao longo de mais de 16000 observações, houve apenas 4 divergências causadas pelas funções de Blackman e de Hamming. A diferença entre ambas, invisível a olho nu, foi de $0,0245$ ponto percentual.

Similarmente, a duração da janela afetou muito pouco as predições. Contudo, isto provavelmente se deve às janelas usadas aqui serem amplamente utilizadas na literatura. Com mais valores, a expectativa é de que haja maior correlação entre duração da janela e taxa de sucesso.

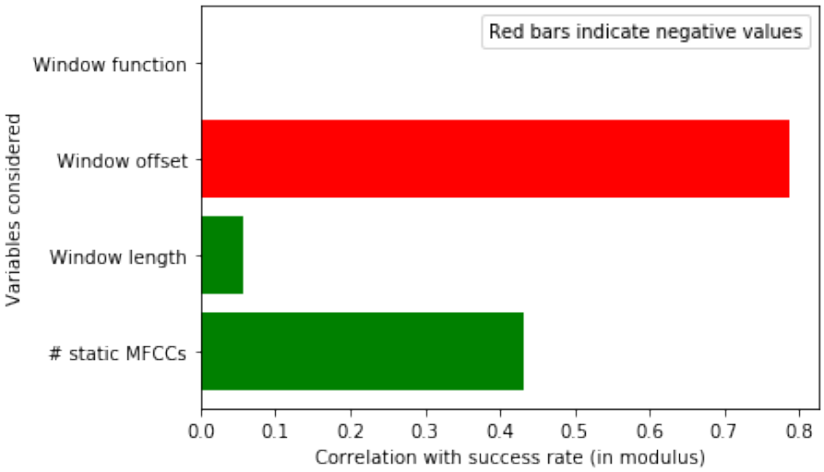

Figura 3: Módulo da correlação linear dos parâmetros com a taxa de sucesso.

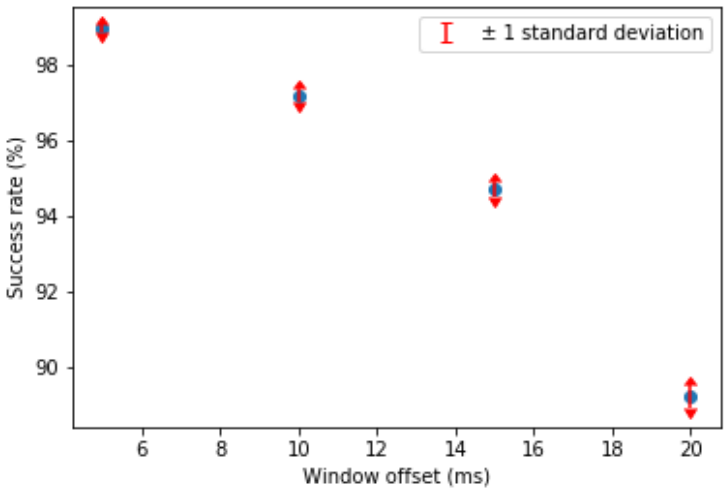

Figura 4: Taxa de sucesso de acordo com o *offset* da janela.

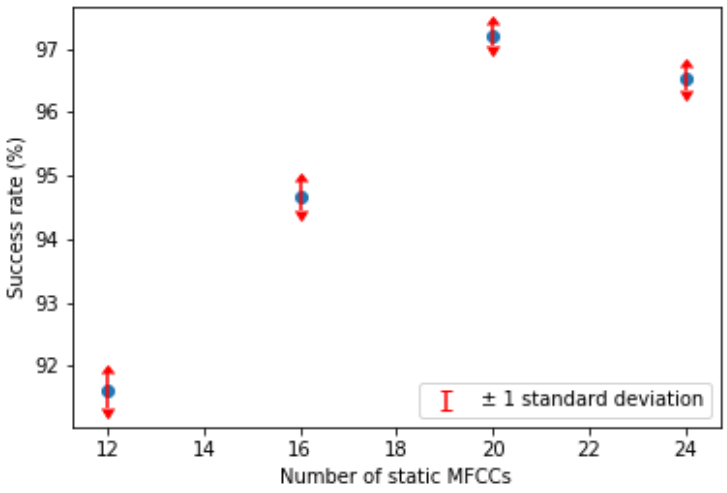

Figura 5: Taxa de sucesso em função da quantidade de coeficientes estáticos utilizados.

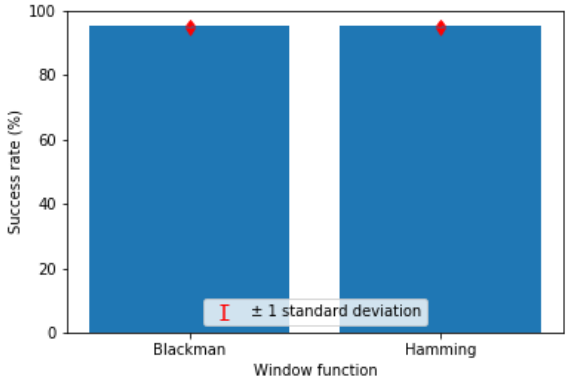

Figura 6: Taxa de sucesso de acordo com a função de janelamento utilizada.

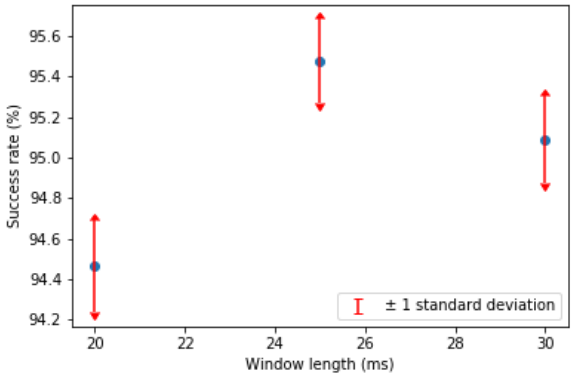

Figura 7: Taxa de sucesso de acordo com a duração da janela.