DeepBach: Harmonizações dos Corais de Bach

Gaëtan Hadjeres, François Pachet, Frank Nielsen

September 25, 2017

Contexto

- Estudo sobre corais do Bach
- Bach tem qu 389 peças de corais
  - um coral é composto por 4 vozes: soprano, contralto, tenor, baixo
  - composições contrapontísticas
  - https://www.youtube.com/watch?v=HPN88O-LX70

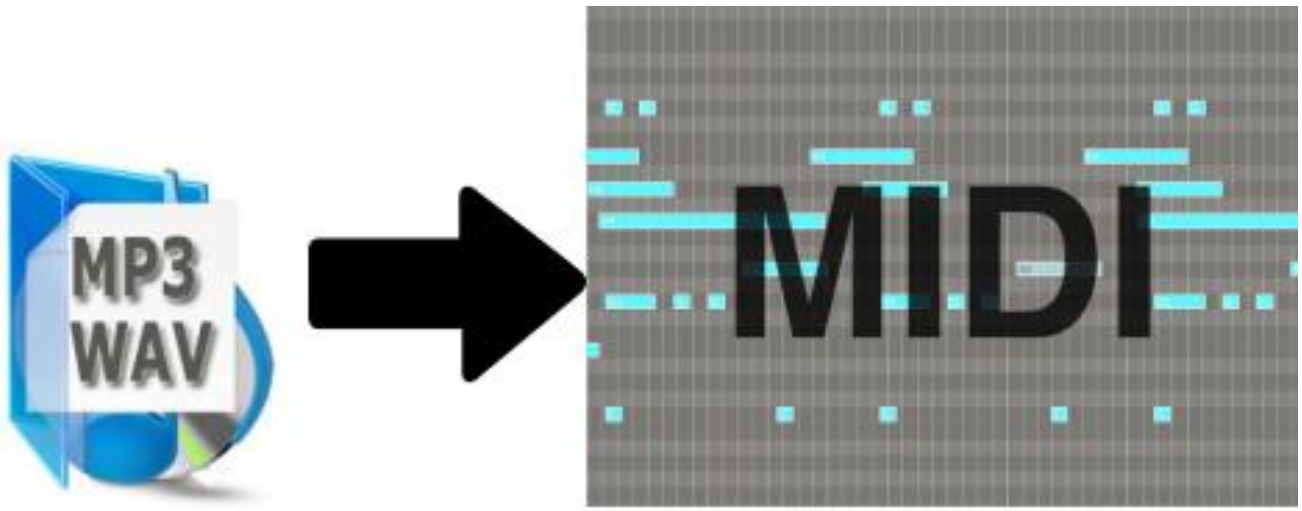
Questões norteadoras

- Será possível imitar o estilo de Bach e gerar músicas automaticamente?
- Como fazer para modelar esses dados?

Análise de dados na música em 3 passos

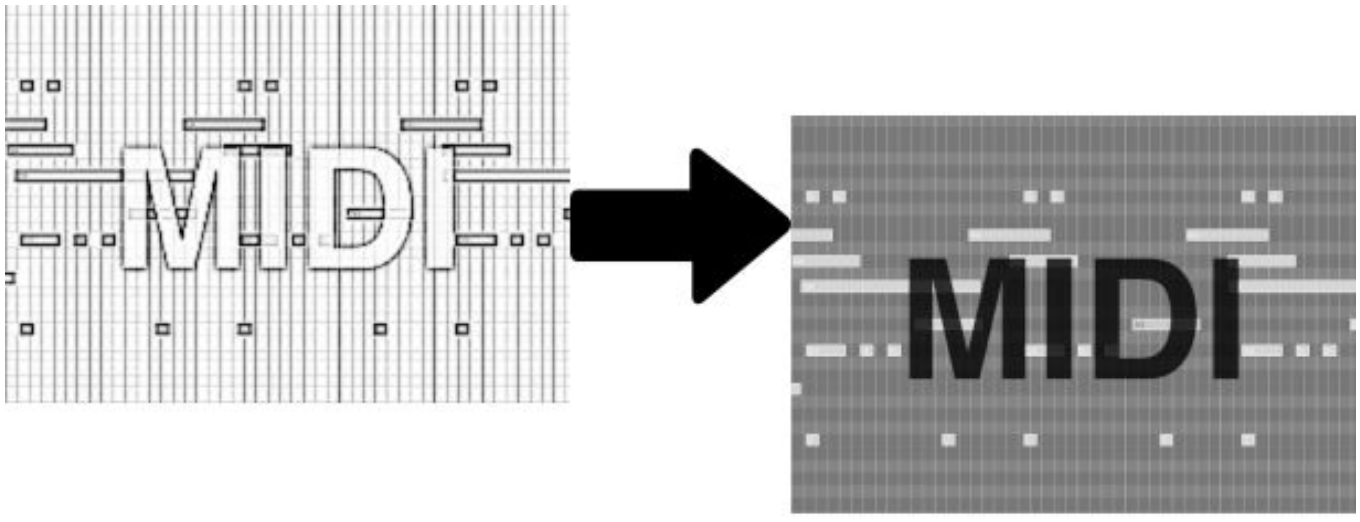
Análise de dados na música em 3 passos

Leitura: extrair dados estruturados dos áudios. - *Music Information Retrieval* - MRI



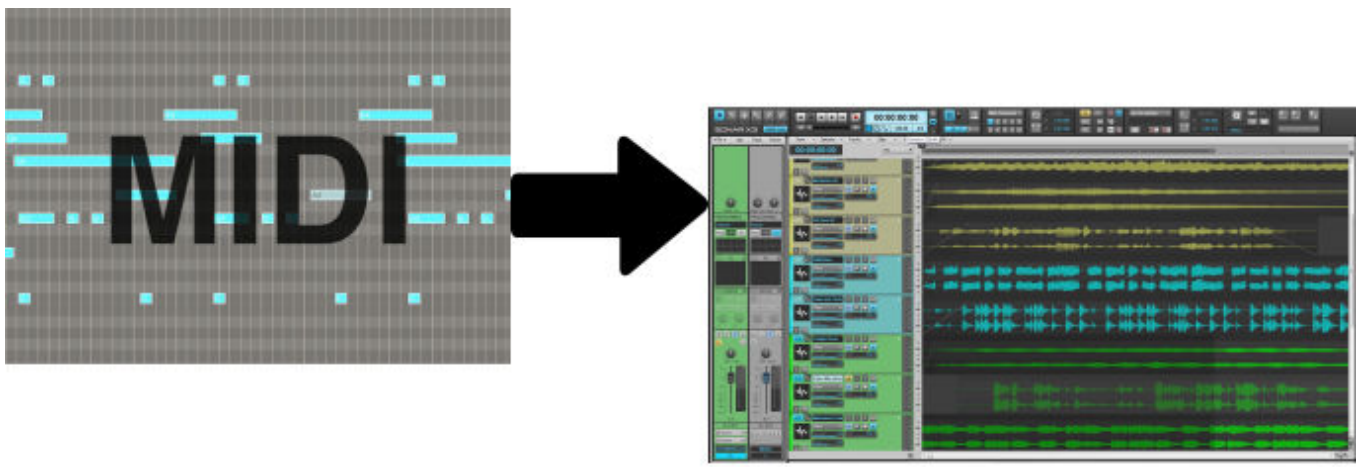
Análise de dados na música em 3 passos

Composição: Criar ou completar músicas automaticamente - Harmonização



Análise de dados na música em 3 passos

Edição: Adicionar efeitos na música para produção



Interesse

Parte 2: composição

Queremos

- Compor uma música nova a partir de um modelo treinado com Bach
- Completar uma melodia existente com o estilo do Bach

Notação

Restrições

- Discretização:
  - intervalo mínimo: semicolcheia $= 1/4$ de batida
  - tempo único
- Corais
  - Apenas 1 melodia por voz
  - Sempre 4 vozes
- Notas

$$\mathcal{V}_i = \{\mathcal{V}_i^t\}_t, t \in \{1, \ldots, T\}, i = \{1, 2, 3, 4\}$$

- T: duração da música
- i: índice da voz

Notas

- quais notas?

$$\mathcal{V}_i^t \in \{\_\_, C_1, C\#_1, D1, D\#_1, ...B_1, C_2, C\#_2, ...\}$$

- cada voz tem um intervalo de notas possíveis (amplitude vocal)

Metadados

$$\mathcal{M} = (\mathcal{S}, \mathcal{F}, \mathcal{R})$$

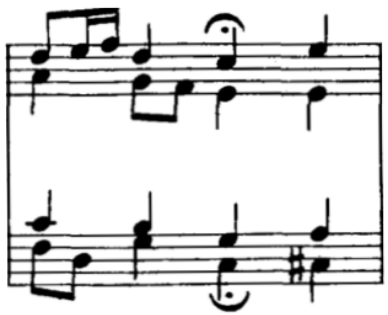
- $\mathcal{S}^t \in \{1, 2, 3, 4\}$: posição da batida
- $\mathcal{F}^t \in \{\text{sim}, \text{não}\}$: fermata
- $\mathcal{R}^t \in \{-7, -6, \ldots, 0, 1, \ldots, 7\}$: guarda o tom, para anotar modulações corretamente e trabalhar com enarmonias.

Dados

Para cada música, nossos dados de entrada são a tupla

$$(\mathcal{V}, \mathcal{M}),$$

em que $\mathcal{V} = (\mathcal{V}_i), i \in 1, 2, 3, 4$.



```
D5,__,E5,F5,D5,__,__,__,C5,__,__,__,E5
A4,__,__,__,G4,__,F4,__,E4,__,__,__,E4
C4,__,__,__,B3,__,__,__,G3,__,__,__,A3
F3,__,D3,__,G3,__,__,__,C2,__,__,__,C#2
 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1
 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0
```

Modelo

Construímos um modelo que nos dê

$$\left\{ p_i(\mathcal{V}_i^t | \mathcal{V}_{\setminus\{i,t\}}, \mathcal{M}, \theta_i) \right\}, i \in 1, 2, 3, 4; t \in 1, \ldots, T$$

- $\mathcal{V}_{\setminus\{i,t\}}$ são todas as notas na vizinhança de $\mathcal{V}_i^t$, excluindo ela própria
- θ_i vetor de parâmetros do modelo.
- Na prática, acabamos usando

$$\tilde{\mathcal{V}}_{\setminus\{i,t\}} = \{\mathcal{V}_j^s, j \neq i, s \in \{t - \Delta t, \ldots, t - 1, t + 1, \ldots, t + \Delta t\}\}$$

- Nesse trabalho, foi usado $\Delta t = 16$.

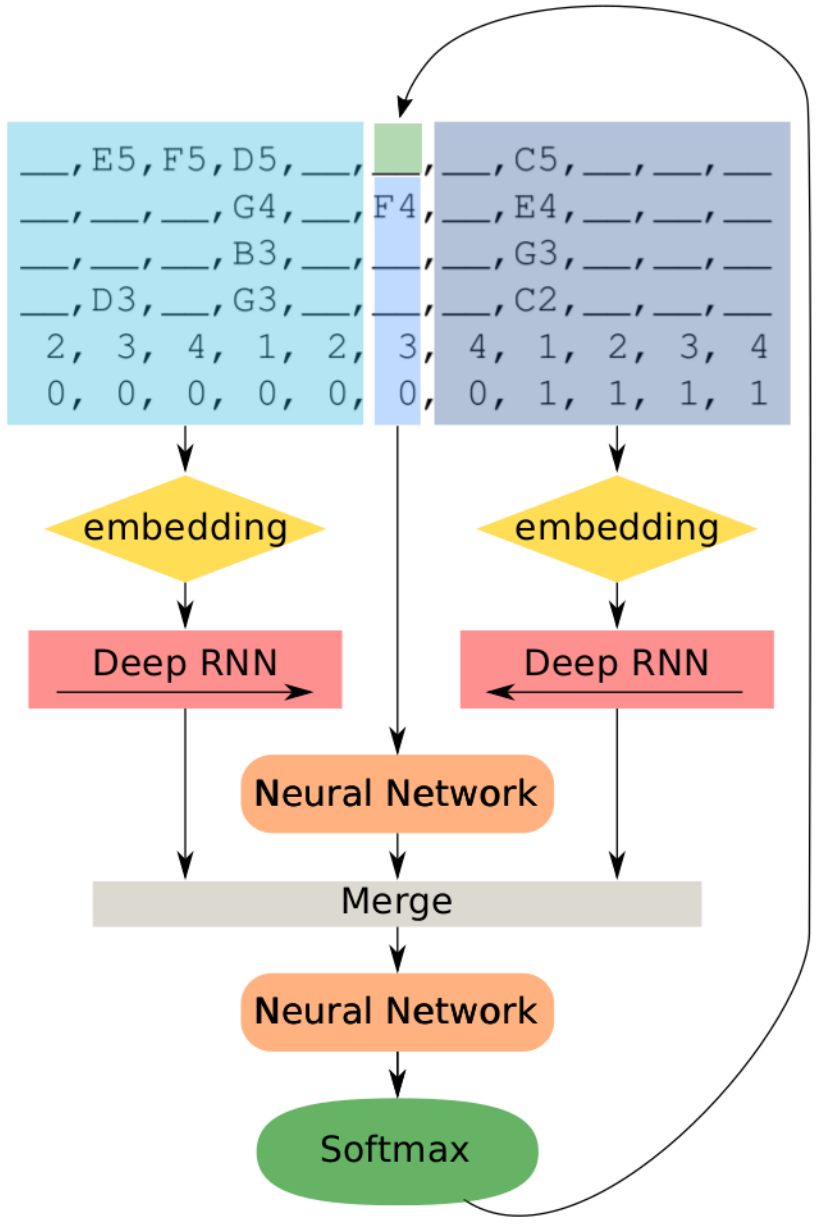
Otimização

Queremos obter θ_i de tal forma que

$$\hat{\theta}_i = \arg\max_{\theta_i} \sum_t \log p_i(\mathcal{V}_i^t | \mathcal{V}_{\setminus\{i,t\}}, \mathcal{M}, \theta_i), \quad i = 1, 2, 3, 4$$

- Para realizar essa otimização, definimos uma arquitetura para o vetor de parâmetros θ_i, utilizando uma rede neural.
- **Vantagens**:
  - Não linearidade considerada de forma flexível
  - Otimização realizada por uma simples descida de gradiente, graças ao *backpropagation*

Arquitetura



Um pouco de NN, RNN e LSTM

Algoritmo generativo (composição)

Algoritmo pseudo-gibbs

- Basicamente, o que fazemos é
  1. Escolher uma voz i e um ponto no tempo t, aleatoriamente: $\mathcal{V}_i^t$
  2. Preencher a vizinhança $\tilde{\mathcal{V}}_{\setminus\{i,t\}}$ com dados do input / iterações anteriores, ou aleatoriamente.
  3. Gerar uma nota com probabilidade $\hat{p}_i$.

Fazemos isso por um número de iterações M definido pelo usuário.

- Não é um Gibbs pois o algoritmo não converge para $p(\mathcal{V})$.
- No entanto, o algoritmo chega a uma distribuição estacionária, segundo os autores.

Resultados

Teste com humanos

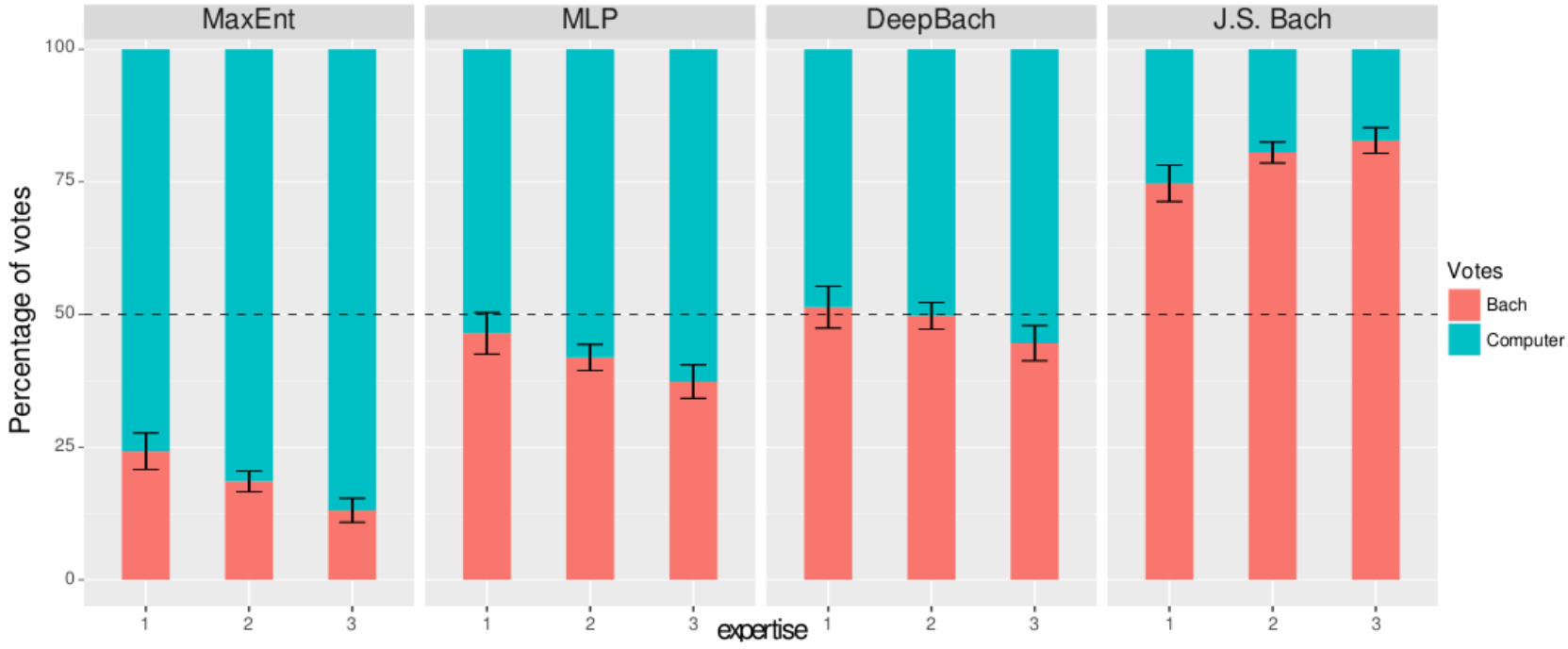
- 1272 pessoas
  - 261 ouvem música clássica raramente
  - 646 adoram música ou praticam música como hobby
  - 365 são estudantes ou profissionais na música
- Comparação de 4 modelos:
  - **MaxEnt**: Rede neural com 1 layer de 1 unidade, sem ativação.
  - **Multilayer Perceptron**: 1 hidden layer com 500 unidades, ativação ReLU
  - **DeepBach**: 200 unidades em cada LSTM, 200 unidades em cada Perceptron, ativação ReLU e dropouts.
  - **Bach**: benchmark

Condução do teste

- Cada pessoa foi submetida a extratos de músicas geradas por algum dos algoritmos (incluindo Bach), chutando se sera uma composição do Bach ou não.
- Os extratos de músicas foram gerados com uma das vozes (soprano) da base de validação, para i) permitir comparação e ii) evitar erros de overfitting.

Resultados

Vermelho: % de pessoas que chutaram “Bach” nas composições



Comentários finais

Conclusões

- É possível fazer composições automáticas do Bach que não ficam horríveis.
- A teoria é razoavelmente simples, a implementação muito simples.
- Trata-se de uma área com muitos avanços recentes e muitas oportunidades.

Futuro (julio)

- Portar esse modelo completamente para o R.
- Fazer um post no blog da curso-r.com sobre esse assunto.
- Implementar diferentes arquiteturas e parametrizações.
- Estudar outros *corpus* para implementar o modelo.

Bibliografia

- https://github.com/Ghadjeres/DeepBach
- https://github.com/felipessalvatore/RNNpresentation