Fichamento

GABRIEL PRÓSPERO REALIZADOR SILVA

Junho 2021

1 DADOS DO ARTIGO

Fast Similarity Matrix Profile for Music Analysis and Exploration
Silva, Diego F. and Yeh, Chin-Chia M. and Zhu, Yan and Batista, Gustavo E. A. P. A. and Keogh, Eamonn
2019

2 RESENHA

A maioria dos algoritmos para análise de música baseada em conteúdo tem em seus núcleos alguma função de similaridade ou distância. Consequentemente, uma variedade de aplicativos dependem de técnicas para avaliar a semelhança entre objetos musicais. Uma abordagem padrão para avaliar a similaridade em gravações musicais é utilizar uma matriz de auto-similaridade (SSM) No entanto, essa representação requer espaço quadrático em relação ao comprimento do vetor de recursos usado para descrever o áudio. Por esse motivo, a maioria dos métodos usados para encontrar padrões na matriz de similaridade são (pelo menos) quadráticos em complexidade de tempo. No entanto, a maioria das informações contidas em matrizes de similaridade é irrelevante ou tem pouco impacto em sua análise. Esta observação sugere a necessidade de uma representação mais eficiente em termos de espaço e tempo das gravações musicais. Em um artigo de conferência anterior dos mesmos autores [9], aplicamos a busca por similaridade de todos os pares de subsequências, também conhecida como junção por similaridade, para avaliar a similaridade entre gravações de áudio para tarefas de MIR. Analogamente às matrizes de similaridade, a representação de toda a junção da subsequência requer um espaço quadrático, e também tem uma alta complexidade de tempo, que depende do comprimento das subsequências a serem unidas. No entanto, demonstramos como explorar uma nova estrutura de dados chamada perfil de matriz, que permite uma representação espaço-eficiente da matriz de junção de similaridade entre subsequências. Além disso, podemos alavancar otimizações recentes na busca de todos os vizinhos para calcular o perfil da matriz de forma eficiente [10].

Definição 3. Junção por similaridade ou junção por similaridade AB. Dadas duas séries de tempo A e B e o comprimento de subsequência desejado m, a

junção de similaridade identifica o vizinho mais próximo de cada subsequência em A de todas as subsequências possíveis de B (ambos com comprimento m). Por meio de uma junção de similaridade, podemos reunir duas informações sobre cada subsequência em A, que são i) a distância até seu vizinho mais próximo em B e ii) a posição de seu vizinho mais próximo em B. Essas informações podem ser armazenadas compactamente em vetores , referido como um perfil de matriz de similaridade (SiMPle) e índice de perfil de matriz de similaridade (índice SiMPle), respectivamente.

Definição 4. . Perfil de matriz de similaridade (SiMPle). Dadas duas séries temporais A e B, SiMPle é uma representação vetorial da junção de similaridade AB. A posição Pi armazena a distância entre a subsequência Ai e seu vizinho mais próximo em B. O índice SiMPle armazena em Ii o índice j da subsequência em Bj que é o vizinho mais próximo de Ai. Quando ambas as séries temporais de entrada se referem à mesma gravação, este é um caso particular de junção por similaridade. Definimos a operação que trata esse caso específico como uma junção de autossimilaridade.

Definição 5. União de autossimilaridade. Dada uma série temporal A e o comprimento de subsequência desejado m, a junção de auto-similaridade identifica o vizinho mais próximo de cada subsequência de A para cada conjunto de subsequência (não trivial) de A. A única grande diferença entre a junção de auto-similaridade (Definição 5) e a junção de similaridade AB (Definição 3) é a exclusão de pares casados trivialmente ao identificar o vizinho mais próximo. A exclusão de correspondências triviais é crucial, uma vez que combinar uma subsequência com ela mesma (ou uma versão ligeiramente deslocada de si mesma) não produz nenhuma informação útil.

O algoritmo original para calcular o SiMPle usa MASS para cada subsequência. Recentemente, pesquisadores notaram que os produtos escalares não precisam ser recalculados do zero para cada subsequência [11]. Em vez disso, podemos reutilizar os valores calculados para a primeira subsequência para fazer um cálculo mais rápido nas próximas iterações. A ideia é aproveitar as interseções entre os produtos requeridos em iterações consecutivas.

Avaliamos nosso método em diferentes cenários em relação aos estilos musicais e ao tamanho dos bancos de dados. Especificamente, avaliamos o SiMPle-Fast em gravações populares e clássicas. O primeiro banco de dados considerado é o YouTube Covers [12], que consiste em 50 composições diferentes, cada uma contendo sete gravações diferentes obtidas de vídeos do YouTube. Os dados originais possuem partições de treinamento e teste, nas quais o conjunto de treinamento possui uma gravação original em estúdio e uma versão ao vivo realizada pelo mesmo artista. Para a tarefa de reconhecimento de música cover, seguimos a mesma configuração de [12] para permitir comparações com os resultados da literatura. O segundo conjunto de dados que consideramos é a coleção amplamente usada de Mazurkas de Chopin [13]. O conjunto de Mazurkas utilizado nesta obra contém 2.919 gravações de 49 peças para piano. O número de gravações de cada música varia de 41 a 95. A avaliação das diferentes escolhas de conjuntos de recursos não é o foco deste artigo. Por esse motivo, corrigimos o uso de recursos baseados em croma em nossos experimentos. Esse tipo

de recurso é predominante em muitas tarefas de recuperação de música e mineração de dados. Especificamente, usamos a estatística normalizada de energia de croma (CENS) [14] para fornecer invariância de tempo local. Esses recursos contam com a aplicação de uma janela Hanning em janelas consecutivas de croma "bruto" e recursos agregados entre a janela para suavizar a variação de croma temporal e reduzir a dimensionalidade dos exemplos.

apresentamos um experimento para avaliar como o SiMPle-Fast melhora o tempo de execução para calcular esses primitivos. A duração da série temporal e o número de trilhas no banco de dados afetam diretamente o tempo de execução de nosso método, bem como qualquer outro método na literatura. Por esse motivo, experimentamos cada parâmetro em um experimento separado.

Os resultados mostram um comportamento linear dos três métodos em relação ao tamanho do conjunto de dados. Em qualquer um desses casos, o método proposto é mais rápido que os outros. Especificamente, para o caso de 10 CENS por segundo (cerca de 1.600 recursos por música em média), o SiMPle-Fast é 8,5 vezes mais rápido do que o DTW e 6,6 vezes mais rápido do que a implementação original do SiMPle. No caso de usar 0,5 CENS por segundo, resultando em 132 CENS por música em média, nosso método é aproximadamente 2,7 vezes mais rápido do que o DTW e 3,3 vezes mais rápido do que o SiMPle. Estes são os menores e maiores números de recursos avaliados em nossos experimentos. O valor adequado para este parâmetro depende da tarefa e dos dados. Para a tarefa de reconhecimento de música cover, descobrimos que 2 CENS por segundo era o valor mais adequado (c.f. Seção III-C). Mais importante, esses resultados mostram que quanto mais longa a série temporal, mais significativa é a melhoria fornecida por nosso método.

3 PLAVRAS-CHAVES

- SIMPLe
- SIMPLe-Fast
- SSM