COMPARAÇÃO DE MODELOS ESTADO DA ARTE PARA A EXTRAÇÃO DA MELODIA DE MÚSICAS EM MIDI

André Luiz Moreira Dutra
Universidade Federal de Minas Gerais
andre.dutra@dcc.ufmg.br

Vitor Rodarte Ricoy
Universidade Federal de Minas Gerais
vitor.ricoy@dcc.ufmg.br

RESUMO

A tarefa de identificar linhas melódicas em músicas polifônicas é um tópico de pesquisa conhecido dentro dos domínios simbólicos e de áudio. A importância desse tópico atraiu o interesse de pesquisadores com foco na Recuperação de Informação Musical e aplicações da musicologia, atingindo bons resultados com um objetivo comum de aplicações industriais. Distinguir a melodia do restante do material em uma partitura escrita pode ser uma tarefa desafiadora, mas o desenvolvimento de métodos usando Deep Learning têm apresentado bons resultados. Neste trabalho, realizamos uma análise comparativa de desempenho entre três modelos para essa tarefa, sendo dois deles estado da arte: LStoM e MidiBERT, e um deles um modelo mais simples utilizado como *baseline*, o Skyline. Os três modelos foram avaliados nas bases Computer Music Analysis Dataset, Orchset e Bach Doodle. Ambos os modelos estado da arte, que envolvem métodos de Deep Learning, foram treinados na base POP909. Os resultados apontam para o MidiBERT como o modelo apresentando os melhor resultado levando em conta todas as bases. Ambos os modelos estado-da-arte apresentaram um ligeiro viés de treino, apresentando desempenho pior para músicas de gêneros diferentes do pop. Apesar disso, o MidiBERT se consolida como o modelo de melhor desempenho mesmo entre as variadas bases.

1. INTRODUÇÃO

A percepção da organização nota a nota da música nos leva à definição de estruturas como frases e contornos melódicos. A identificação do que é percebido como melodia pelos ouvintes em uma música emergiu recentemente como um tópico importante na comunidade da Recuperação da Informação Musical [1].

Uma linha melódica incorpora propriedades musicais que são ricas de informação. A estrutura e ritmo da melodia carrega consigo características expressivas da perspectiva do compositor ao construir uma música, assim como a perspectiva do intérprete, ao executá-la.

Assim, ser capaz de extrair a linha melódica de uma música com uma boa precisão pode revelar informações relevantes, que podem auxiliar outras linhas de pesquisa como analisar um estilo de composição ou classificar as tendências de um intérprete, por exemplo. A extração de melodias com uma boa precisão também pode se mostrar relevante para melhorar os resultados de algoritmos de busca ou de recomendação aplicados a músicas, assim como melhorar sistemas de geração de músicas.

Ao considerar ambos os domínios simbólicos e de áudio, a tarefa de recuperar as notas da melodia de uma música polifônica não é trivial, até mesmo para humanos. Por exemplo, ao ouvir uma música algumas pessoas apresentam dificuldade de distinguir intervalos de nota, podendo julgar intervalos como sendo uma terça maior ao ouví-lo sozinho, e como uma quarta ao ouví-lo dentro do contexto de uma música [2]. Já durante o ato de ler uma partitura, ou renderizá-la com apenas um timbre, os intervalos de notas de vozes diferentes podem ser percebidos como intercambiáveis. Esses dois desafios inspiraram a pesquisa no campo de extração de vozes, que consiste em dividir uma música polifônica em um conjunto de melodias monofônicas [3] [4].

Nesse trabalho, iremos focar na tarefa de identificar a linha melódica principal em uma partitura simbólica, no formato MIDI. Os trabalhos nessa tarefa podem ser divididos em duas categorias. A primeira, que não está no escopo desse trabalho, são modelos que identificam a faixa que contém a melodia da música, recebendo como entrada um arquivo MIDI em que sabidamente uma das faixas é a própria melodia. já separada. A outra categoria, que será abordada nesse trabalho, são modelos que identificam a linha melódica sem qualquer conhecimento prévio ou separação de faixas. O foco na segunda categoria se dá por ela representar um problema mais flexível e com mais casos de uso no mundo real.

Para avançar ainda mais no campo da extração de melodia, nesse trabalho iremos comparar dois modelos estado da arte, que se utilizam da abordagem de Deep Learning, assim como um algoritmo clássico para extração de melodia, como *baseline*. Tais modelos serão testados em diferentes bases, sintéticas e reais, com o objetivo de analisar o resultado deles em bases diferentes da base para a qual eles foram treinados.

O artigo apresentado na ISMIR 2022, em que o modelo LSToM [5] é definido, foi utilizado como base para esse trabalho desenvolvido, por apresentar uma coletânea

de métodos estado da arte para essa tarefa, além de propor o modelo LSToM.

Com base nos resultados alcançados nesse artigo, primeiramente escolhemos dois modelos para serem comparados: o próprio LSToM e o MidiBERT [6]. Tal escolha foi feita pelos dois primeiros modelos terem sido treinados na base POP909 [7] e pelo MidiBERT ter apresentado um resultado um pouco melhor que o LSToM nessa base, enquanto os dois métodos dominam todos os outros métodos explorados no artigo [5]. Também iremos incluir na comparação o algoritmo Skyline [8] [9], por se tratar de um algoritmo clássico para essa tarefa, que não envolve Deep Learning.

É importante notar que existem modelos para extração de melodias que utilizam a imagem do Piano Roll gerado pelo arquivo MIDI e, a partir dessas imagens, treinam uma rede convolucional para classificar as melodias. Tal abordagem apresenta bons resultados, mas não foi considerada nesse trabalho por apresentar resultados inferiores às duas abordagens de Deep Learning escolhidas.

Os três modelos foram testados nas bases Computer Music Analysis Dataset [10], Orchset [11] e Bach Doodle [12]. Sendo a primeira delas uma base com algumas músicas conhecidas, a segunda uma base de músicas clássicas tocadas por orquestra e a terceira um conjunto de melodias acompanhadas por acordes gerados por uma IA. Em todas as bases, os modelos foram executados para as músicas completas em MIDI, e a resposta de cada modelo, também em MIDI, foi comparada com o arquivo MIDI contendo a melodia correta da música.

Com a execução dos modelos obtivemos as métricas de erro deles em cada uma das bases e esse resultado foi utilizado para comparar o desempenho dos modelos de Deep Learning em bases diferentes da base na qual eles foram treinados, validados e testados, enquanto também consideramos os resultados de um algoritmo clássico, como *baseline*.

2. MODELOS ESCOLHIDOS

Primeiro iremos detalhar cada um dos modelos escolhidos, descrevendo a sua ideia de funcionamento. Para maior detalhes de implementação, é recomendado ler o artigo original em que cada modelo foi proposto.

2.1 Skyline

Na literatura, o algoritmo de Skyline [8] [9] é a abordagem clássica mais popular para se extrair a melodia de músicas polifônicas.

O algoritmo, descrito em Algoritmo 1, basicamente agrupa todas as notas do arquivo MIDI em apenas um canal e seleciona as notas mais agudas, ou seja, com o tom mais alto. No caso de termos múltiplas notas com o mesmo tempo de *onset*, a nota com a maior frequência é mantida, enquanto o restante das notas são eliminadas. O algoritmo também pode diminuir a duração de notas, quando uma nova nota com a frequência mais alta ocorre no meio da execução de uma nota que, em seu início, era a mais aguda.

Algoritmo 1 Skyline

$notes \leftarrow rawNotes$
$skylineNotes \leftarrow []$
for $note \in notes$ **do**
  $nextNote \leftarrow$ notas com o mesmo início $note$
  if $note \notin importantNotes$ and $note = argmax(pitch(sameTimeNote))$ **then**
    $nextNote \leftarrow$ próxima nota com tempo de início diferente
    $note.end = min(note.end, nextNote.start)$
    Adiciona $note$ em $skylineNotes$
  end if
end for

Podemos tecer três grandes críticas ao Skyline. A primeira consiste no fato da alteração da duração das notas potencialmente altera a melodia. A segunda critica tem relação com o fato de todos os canais serem mesclados, o que pode remover intervalos de silêncio que faziam parte da melodia. Por fim, a terceira crítica consiste na observação de que é possível que tenhamos notas de acompanhamento mais agudas do que as notas da melodia.

2.2 MidiBERT

O modelo MidiBERT [6] foi treinado para quatro tarefas relacionadas ao domínio simbólico das músicas: extração de melodia, predição de velocidade, classificação do compositor e classificação de emoção. O modelo construído pelo MidiBERT para a extração de melodia classifica cada evento de nota do MIDI em melodia, ponte ou acompanhamento.

Inspirado pela tendência crescente de tratar música em MIDI com uma linguagem, em modelos de Deep Learning, o MidiBERT aplica uma rede baseada em Transformers, usando uma estratégia de treinamento auto supervisionada, chamada "Masked Language Modeling", que é largamente usada em modelos BERT-like no Processamento de Linguagem Natural.

Apesar da fama dos modelos BERT-like, existem poucos trabalhos que utilizam tais modelos em tarefas relacionadas a músicas, mais especificamente no domínimo simbólico.

A abordagem utilizada pelo MidiBERT considera que, de forma similar a textos, uma música em MIDI pode ser considerada como uma sequência de eventos musicais, ou *tokens*. No entanto, o que faz com que a música seja diferente do texto é o fato das notas musicais estarem associadas com um fator temporal, como a duração da nota, além de que múltiplas notas podem ser tocadas ao mesmo tempo. Assim, para representar músicas, precisamos de *tokens* que tenham descrições relacionadas às notas, como o tom e a duração das notas, assim como *tokens* relacionados às métricas, distribuindo as notas temporalmente.

Para construir os *tokens* o MidiBERT utiliza duas abordagens a *REMI Token Representation* e a *Compound Word*. A primeira delas codifica separadamente em um *tokens* as informações de *Bar*, *Sub-beat*, *Pitch* e *Duration*, sendo que

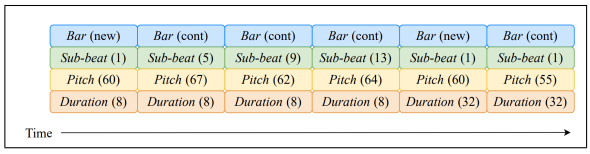

Figura 1. *Tokens* na representação *Compound Word*, utilizada pelo MidiBERT.

cada uma delas ocorre separadamente seguindo a sequência de *Sub-beat*, *Pitch* e *Duration* dentro de uma mesma *Bar*.

Já a Compound Word, vista na Figura 1, adota uma representação mais densa, em que cada *token* contém informações de *Sub-beat*, *Pitch* e *Duration*, além de conter a informação do *Bar*, indicando se é um novo (*new*) ou uma continuação do atual (*cont*).

O artigo original compara as duas representações, mas para esse trabalho utilizamos a representação *Compound Word*. Isso traz uma vantagem ao alimentar o Transformer com uma concatenação dos atributos relevantes da nota, em cada *token*, permitindo também que o Transformer faça a predição de cada um dos atributos com um *head* diferente na saída.

Dessa forma, a representação *Compound Word* possui ao menos duas vantagens ao compará-la com a *REMI Token Representation*. A primeira vantagem acontece devido os passos necessários para representar uma música são reduzidos, já que os tokens são mesclados em um *super token* (ou uma *compound word*). Outra vantagem, são os mecanismos de *self-attention* do Transformer operarem sobre o *super token*, o que é mais significativo, visto que tais *tokens* representam diferentes aspectos de uma nota musical de forma independente.

2.3 LSToM

O modelo LSToM [5] é um modelo supervisionado bidirecional do tipo *Long-Short Term-Memory* (*biLSTM*), que é um tipo de Rede Neural Recorrente, com 6 camadas. O modelo foi chamado de LSToM, significando *Large Score to Melody*.

O modelo recebe um conjunto de *features* derivadas do arquivo MIDI como entrada e classifica quais notas fazem parte da linha melódica principal. Para calcular essas *features*, os eventos de nota, presentes no arquivo MIDI, são adaptados para representar as diferentes características que serão capturadas pelo modelo.

O modelo captura as seguintes características:

- *pitch*: Representa o tom da nota, tendo o valor igual ao número da nota na codificação do MIDI.
- *dur*: Representa a duração da nota, expressa no valor da colcheia equivalente;
- *pitch_dist_below*: Representa a distância absoluta da nota até a nota mais grave tocada simultaneamente, expresso em semitons;
- *pitch_dist_above*: Representa a distância absoluta da nota até a nota mais aguda tocada simultaneamente, expresso em semitons;
- *pos_in_bar*: Representa a batida em que a nota é tocada dentro compasso atual, expresso no valor da colcheia equivalente;
- *pitch_in_scale*: Indica se o tom da nota está dentro da escala diatônica da música, expresso com um valor booleano.

Assim, o modelo alimenta a rede com um vetor de característica para cada uma dos eventos de nota do arquivo MIDI, na ordem em que eles aparecem no arquivo. Os detalhes das camadas do modelo em si não são melhor detalhadas no artigo, mas seguem a ideia base das redes *Long-Short Term-Memory*.

É importante notar que o modelo é consideravelmente menos complexo, ao ser comparado com o MidiBERT, já que o MidiBERT tem 14 camadas e 111.298.052 parâmetros, enquanto o LSToM possui 6 camadas e 875.462 parâmetros.

3. BASES ESCOLHIDAS

Nessa seção serão detalhadas cada uma das bases escolhidas para comparar os modelos. Para acesso aos *datasets* em si, assim como descrições mais detalhadas, deve-se verificar os artigos ou *websites* em que as bases foram publicadas.

3.1 Computer Music Analysis Dataset

A base Computer Music Analysis Dataset [10] fornece dados de treino e teste com labels de melodia, acordes e estrutura. São disponibilizadas 52 músicas para treino e 22 para teste.

Focando nas informações de melodia, que são relevantes para esse trabalho, temos que os dados de treino têm a melodia concentrada em uma única faixa do MIDI. Assim, temos um arquivo que indica qual faixa contém a melodia de cada arquivo.

Já nos dados de teste, temos apenas uma marcação que indica, para cada compasso, qual canal (ou canais) no qual a melodia está contida. Dessa forma, é possível construir o arquivo MIDI com a melodia ao analisar cada compasso e filtrar os canais que contém melodia.

Esse *dataset* não foi proposto em um artigo, portanto a sua única referência é o *website* no qual ele está hospedado. Isso o torna uma base de dados menos relevante, mas devido à escassez de bases, aliado ao fato desse *dataset* conter músicas reais, ele foi selecionado para a avaliação.

3.2 Orchset

A base Orchset [11] é uma base mais completa, desenvolvida para o desenvolvimento e avaliação de algoritmos de extração de melodia. Ela foi publicada em um artigo,

que detalha os seus passos de construção, já que a sua criação envolveu um trabalho mais complexo e multidisciplinar.

A base contém 64 trechos de músicas tocadas por orquestras sinfônicas, acompanhados da anotação de melodia corresponde de cada trecho, assim como um arquivo MIDI dessa melodia. A melodia, para os autores da base, foi definida como "uma sequência monofônica de notas que um ouvinte provavelmente irá reproduzir ao ser pedido para assobiar ou murmurar uma música polifônica". Assim, a base foca na percepção humana da melodia.

A criação da base envolveu a seleção dos trechos, sessões de gravação de pessoas cantando juntamente dos trechos, análise das gravações e anotações da melodia.

O maior problema dessa base para a nossa aplicação foi o formato das músicas polifônicas, que foi apresentado em WAV. Assim, convertemos as músicas WAV em MIDI, utilizando a ferramenta *Basic Pitch* [13], que é uma ferramenta de código aberto desenvolvida pelo Spotify. Essa ferramenta utiliza um modelo estado da arte e foi usada com o objetivo de obter a melhor representação dos arquivos WAV no formato MIDI.

Entretanto, essa conversão não é perfeita, assim consideramos que essa base de dados é uma base de dados sintética, já que o dado de entrada, que era um dado real, foi distorcido pela conversão para MIDI, por meio do modelo do *Basic Pitch*.

Apesar disso, essa base é relevante e apresenta um estilo de música consideravelmente diferente da base em que os modelos foram treinados, já que os modelos foram treinados com músicas populares e a base apresenta músicas clássicas.

3.3 Bach Doodle

A base Bach Doodle [12] é uma base sintética criada a partir dos dados coletados por um *Doodle* disponibilizado na página principal do buscador da Google do dia 21 de Março de 2019, data do 334º aniversário do compositor Johann Sebastian Bach, até o dia 23 de Março de 2019. O *Doodle* recebeu mais de 55 milhões de interações, das quais pouco mais de 21,6 milhões foram incluídas em um *dataset* disponibilizado pela Google. Vale notar que das 21,6 milhões de melodias, temos 14 milhões de melodias únicas.

No *Doodle* os usuários criam suas próprias melodias, por meio de uma interface desenvolvida para ser fácil de utilizar e para representar uma partitura. Quando os usuários inserem uma melodia na interface, a melodia é harmonizada por um modelo de aprendizado de máquina, chamado Coconet [14]. O modelo cria uma harmonia para a melodia seguindo o estilo do compositor Bach.

Uma observação interessante sobre o processo implementado pelo *Doodle* é o fato dele executar o modelo localmente, na máquina do usuário, ou nos servidores do Google, caso a máquina do usuário não tenha capacidade para executá-lo. Também é importante notar que ao ouvir a harmonização gerada, o usuário tinha a opção de avaliar o resultado e podia escolher se a composição gerada seria inserida no *dataset* que utilizamos.

As mais de 21,6 milhões de composições do *dataset* estão divididas em 8,5 milhões de sessões. Cada sessão é uma interação anônima de um usuário com o *Doodle*, sendo que cada interação pode gerar múltiplas composições. Cada composição consiste da melodia de entrada do usuário, em MIDI, a harmonização de quatro vozes gerada pelo Coconet, também em MIDI, assim como os metadados: o país de origem, a nota do usuário para o resultado, o tom da composição, o tempo gasto para compor a melodia e o número de vezes que o usuário ouviu a composição retornada.

Essa base foi utilizada nesse trabalho por sua larga escala, assim como sua variedade de melodias, já que as melodias foram criadas pelos usuários, abordando inúmeros estilos musicais, incluindo estilos regionais. Entretanto, assim como a base anterior, essa é uma base sintética, já que a harmonização foi desenvolvida por um modelo de aprendizado de máquina.

Também vale notar que essa base possui um viés do estilo do Bach, já que a harmonização foi criada para se assemelhar às características desse compositor. Assim, mesmo que as melodias apresentadas na base sejam de diferentes estilos musicais, a música gerada no final apresenta fortes tendências à música clássica, já que o seu acompanhamento foi feito com um modelo que captura o estilo de um compositor clássico.

4. METODOLOGIA

4.1 Base de Dados

Primeiramente, realizamos um tratamento nas bases de dados. Para a base Computer Music Analysis Dataset [10], geramos um script em Python para extrair a melodia de cada música, de acordo com o processo descrito na página da base. Com isso, obtemos os arquivos MIDI com as músicas completas e os arquivos MIDI das melodias, para cada música da base.

Já para o Orchset [11], passamos cada um dos arquivos de entrada pelo modelo *Basic Pitch* [13], para converter as entradas de WAV para MIDI. Como a base já disponibiliza as melodias esperadas em MIDI, após a conversão das entradas obtemos os arquivos MIDI com as músicas completas e os arquivos MIDI das melodias, para cada música da base.

Por fim, para o Bach Doodle [12], precisamos extrair os arquivos MIDI de entrada e saída, fornecidos no formato *JSONL* ou *TFRECORD*. Ao extrair os arquivos, o MIDI de saída foi utilizado como entrada, ou seja, como o MIDI da música completa, e o MIDI de entrada foi utilizado como saída esperada, ou seja, como o MIDI da melodia da música.

4.2 Avaliação dos Modelos

Para a avaliação dos modelos primeiro precisamos definir uma implementação a ser utilizada na avaliação, para cada um deles.

Para o LSToM [5] utilizamos o código fonte disponibilizado pelo próprio autor do artigo que define o modelo. Tal código fonte precisou de pequenas modificações para funcionar corretamente, mas com ele conseguimos treinar o modelo no *dataset* POP909, assim como foi feito no artigo em que o modelo foi proposto. Após o treinamento, podemos executar a predição do modelo nas bases selecionadas para a avaliação.

Já para o modelo MidiBERT [6], utilizamos o código fonte disponibilizado pelos autores do artigo, assim como o modelo já treinado na base POP909, também disponibilizado pelos autores. Para a execução da predição fizemos pequenas alterações na lógica de entrada e saída dos dados, mas a predição em si foi executada sem modificações, em constraste com o LSToM.

Por fim, para o Skyline, implementamos o algoritmo proposto no Algoritmo 1. Como o algoritmo não exige treinamento e nenhuma forma mais complexa de tratamento de dados, a sua implementação e execução foi mais simples.

Após a execução de cada um dos modelos para todas as bases, obtemos a saída dos modelos, em MIDI. Tais arquivos MIDI foram usados, em conjunto com os arquivos MIDI das melodias esperadas para cada base, para calcular as métricas de erro de cada modelo em cada base.

A métrica escolhida foi a mesma métrica utilizada no artigo em que o MidiBERT [6] foi proposto, pois ao procurarmos por métricas de comparação de dois arquivos MIDI de melodia, a métrica utilizada pelo MidiBERT pareceu a métrica mais robusta e que captura mais informações relevantes em seu cálculo.

Por fim, os resultados foram agregados por modelo e por métrica, considerando todas as bases, e também a acurácia foi agregada para cada modelo e cada base. Tais resultados serão apresentados e analisados na seção a seguir.

5. RESULTADOS

Algoritmo	Acurácia	Precisão	Revocação	F1
Skyline	0.86	0.13	0.47	0.21
LStoM	**0.95**	0.49	0.13	0.21
MidiBERT	**0.95**	**0.61**	**0.69**	**0.65**

Tabela 1. Acurácia, precisão, revocação e F1 de cada modelo dos modelos na previsão de melodias em todas as bases. Em negrito está destacado o maior valor para cada métrica.

A Tabela 1 descreve os resultados gerais de acurácia, precisão, revocação e f1 para os três modelos analisados, em relação a todas as bases.

As métricas foram calculadas considerando o desempenho de cada modelo na classificação nota-a-nota de cada melodia prevista em relação à melodia esperada, conforme [6]: para cada nota presente na melodia prevista e na esperada, temos um verdadeiro positivo. Notas na melodia prevista e não na esperada constituem falsos positivos. Notas presentes na melodia esperada e não na melodia prevista constituem falsos negativos, e notas ausentes em ambas constituem verdadeiros negativos. O somatório dos falsos e verdadeiros positivos e negativos de todas as instâncias de todas as bases foi utilizado para o cálculo do desempenho de cada modelo.

Primeiramente, é importante observar que, embora os três modelos apresentem acurácia acima de 80%, o Skyline e o LStoM apresentaram precisão, revocação e F1 consideravelmente baixos. Isso ocorre devido ao desbalanceamento da base quando interpretada como uma base de classificação nota-a-nota.

Neste caso, em um mesmo arquivo existem muito mais notas de acompanhamento e transições do que de melodias. Com isso, um modelo que não prevê a melodia bem ainda pode ter alta acurácia desde que sua saída seja consideravelmente menor que a quantidade de notas total da entrada.

É notável que o modelo MidiBERT teve um melhor desempenho num geral, apresentando maior valor para todas as métricas. De maneira semelhante, o LStoM também teve um resultado melhor que o Skyline, o algoritmo determinístico utilizado de *baseline* para ambas as comparações.

No entanto, ambos o LStoM e o MidiBERT apresentaram desempenho inferior ao documentado em seus respectivos artigos, e mais próximo do Skyline. Isto possivelmente se evidencia pelo fato de que ambos os modelos foram treinados na base POP909, constituída apenas por músicas pop.

Com isso, como dois dos *datasets* testados, Orchset e Bach Doodle, descrevem músicas clássicas, é esperado que o desempenho geral dos modelos caia, conforme documentado em suas respectivas literaturas.

Também podemos atribuir o menor desempenho na base Orchset ao fato dos MIDIs utilizados como entrada nessa base terem sido gerados com base em arquivos WAV. Tal conversão pode levar à perda ou distorção de informação, o que torna mais difícil a predição correta da melodia, tendo como base as entradas convertidas.

Estes resultados corroboram observações na literatura, na qual é atestado que, embora ambos os modelos apresentem desempenho significativamente superior ao *baseline*, esta diferença pode se mostrar menor com a utilização de bases de dados com a natureza distinta da base utilizada para treino [5] [6].

	Bach Doodle	CMAD	Orchset	Pop909
Skyline	**0.86**	0.83	0.80	0.85
LStoM	0.82	0.93	0.84	**0.95**
MidiBERT	0.80	0.92	0.79	**0.98**

Tabela 2. Acurácia de cada modelo na tarefa de previsão de melodias em cada base. Em negrito está destacada a maior acurácia obtida por cada modelo.

Na Tabela 2 são apresentadas as acurácias de cada mod-

elo com relação a cada base de dados testada. Também foram coletados o desempenho de cada modelo para a base POP909, utilizada originalmente no MidiBERT e no LStoM, para utilizarmos o desempenho dos modelos na sua base de treino como referência.

Os resultados observados reforçam a hipótese levantada anteriormente com relação à diferença de desempenho geral dos modelos LStoM e MidiBERT, ao compará-lo com o desempenho documentado em seus respectivos artigos.

Conforme podemos observar na Tabela 2, ambos os modelos apresentaram o melhor desempenho na base POP909, na qual foram treinados, como era esperado. Além disso, o desempenho de ambas foi menor para as bases Orchset e Bash Doodle em relação às demais bases.

Estes resultados apontam para a presença de um viés de treinamento nos modelos LStoM e MidiBERT com relação às suas bases de treino, com ambos performando melhor em bases contendo músicas pop, como é o caso da *Computer Music Analysis Dataset* (CMAD) e da POP909, do que bases contendo músicas clássicas, que é o caso da *Orchset* e da *Bach Doodle*.

Por outro lado, o Skyline manteve seu desempenho relativamente consistente nas quatro bases, apesar da variabilidade dos dados. Deste modo, embora possua um desempenho geral consideravelmente pior, principalmente para as demais métricas, o Skyline se apresenta como uma alternativa mais robusta à variabilidade de dados, devido à consistência de acurácia das suas classificações.

Por fim, vale notar que, ao considerar a acurácia, o LStoM apresenta o melhor resultado geral entre as bases. No entanto, o desempenho geral do MidiBERT, embora mais sensível à variabilidade, ainda supera os demais modelos com relação às outras métricas ded precisão, revocação e f1.

6. CONCLUSÃO

Neste trabalho, realizamos uma análise de desempenho de modelos estado-da-arte na abordagem do problema de extração de melodias de músicas polifônicas representadas simbolicamente, mais especificamente no formato MIDI. Através do uso de bases de diferentes gêneros musicais, principalmente bases contendo músicas pop e músicas clássicas, pudemos observar não só o desempenho geral dos modelos, como também sua robustez relativa à base de dados utilizada.

A partir dos resultados, podemos concluir que o MidiBERT se consolida como o modelo mais efetivo dentre os testados na execução da tarefa proposta. Embora ligeiramente sensível à variabilidade na base, apresentando desempenho melhor para músicas pop, seu desempenho geral foi consideravelmente maior que os demais modelos, principalmente levando em consideração métricas robustas ao desbalanceamento da base.

Mas também devemos destacar que o LStoM, apesar de ter resultados gerais piores ao ser comparado com o MidiBERT, apresenta resultados razoáveis, superiores ao *baseline*. Considerando que o LStoM é um modelo bem menor, e portanto mais rápido de ser executado, ele pode ser mais adequado do que o MidiBERT, a depender da situação.

Como perspectivas futuras, temos a realização de testes com bases maiores e mais diversificadas. Também seria interessante o uso de mais bases com dados reais, já que nesse trabalho utilizamos uma maioria de bases sintéticas, devido à escassez de bases.

Além disso, seria pertinente investigar o desempenho dos mesmos modelos treinados em bases menos enviesadas, ou até mesmo treinados em diversas bases com diferentes gêneros musicais, com o objetivo de obter resultados ainda melhores de ambos.

7. REFERÊNCIAS

[1] J. Salamon, E. Gomez, D. P. W. Ellis, and G. Richard, "Melody extraction from polyphonic music signals: Approaches, applications, and challenges," *IEEE Signal Processing Magazine*, vol. 31, no. 2, pp. 118–134, 2014.

[2] N. Cook, *Music, Imagination, and Culture*, ser. ACLS Humanities E-Book. Oxford University Press, 1990. [Online]. Available: https://books.google.com.br/books?id=5DKiUOw519oC

[3] E. Cambouropoulos, "Voice and stream: Perceptual and computational modeling of voice separation," *Music Perception - MUSIC PERCEPT*, vol. 26, pp. 75–94, 09 2008.

[4] R. de Valk and T. Weyde, "Deep neural networks with voice entry estimation heuristics for voice separation in symbolic music representations," in *19th International Society for Music Information Retrieval Conference (ISMIR 2018)*, May 2018. [Online]. Available: https://dblp.uni-trier.de/db/conf/ismir/ismir2018.html

[5] K. Kosta, W. T. Lu, G. Medeot, and P. Chanquion, "A deep learning method for melody extraction from a polyphonic symbolic music representation," in *Proceedings of the 23rd International Society for Music Information Retrieval Conference*. ISMIR, Dec. 2022, pp. 756–763. [Online]. Available: https://doi.org/10.5281/zenodo.7402960

[6] Y.-H. Chou, I.-C. Chen, C.-J. Chang, J. Ching, and Y.-H. Yang, "MidiBERT-Piano: Large-scale pre-training for symbolic music understanding," *arXiv preprint arXiv:2107.05223*, 2021.

[7] Z. Wang*, K. Chen*, J. Jiang, Y. Zhang, M. Xu, S. Dai, G. Bin, and G. Xia, "Pop909: A pop-song dataset for music arrangement generation," in *Proceedings of 21st International Conference on Music Information Retrieval, ISMIR*, 2020.

[8] A. Uitdenbogerd and J. Zobel, "Melodic matching techniques for large music databases," in *Proceedings of the Seventh ACM International Conference*

on Multimedia (Part 1), ser. MULTIMEDIA '99. New York, NY, USA: Association for Computing Machinery, 1999, p. 57–66. [Online]. Available: https://doi.org/10.1145/319463.319470

[9] ——, "Manipulation of music for melody matching," *Proceedings of the 6th ACM International Conference on Multimedia, MULTIMEDIA 1998*, 08 2002.

[10] Computer Music Project - Carnegie Mellon University, "Computer music analysis dataset," 2019, dados obtidos pelo site do projeto: https://www.cs.cmu.edu/~music/data/melody-identification/.

[11] J. Bosch and E. Gomez, "ORCHSET: a dataset for melody extraction in symphonic music recordings," Jun. 2018. [Online]. Available: https://doi.org/10.5281/zenodo.1289786

[12] C.-Z. A. Huang, C. Hawthorne, A. Roberts, M. Dinculescu, J. Wexler, L. Hong, and J. Howcroft, "The Bach Doodle: Approachable music composition with machine learning at scale," in *International Society for Music Information Retrieval (ISMIR)*, 2019. [Online]. Available: https://goo.gl/magenta/bach-doodle-paper

[13] R. M. Bittner, J. J. Bosch, D. Rubinstein, G. Meseguer-Brocal, and S. Ewert, "A lightweight instrument-agnostic model for polyphonic note transcription and multipitch estimation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Singapore, 2022.

[14] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, "Counterpoint by convolution," 2019.