CD-003

楽器で演奏された単旋律フレーズをクエリとした楽曲検索システム*

松下陸 (法政大学情報科学部), 伊藤克亘 (法政大学情報科学部)

1　まえがき

楽曲検索システムにおいて膨大なデジタル楽曲データベースの中からユーザーが求める楽曲を適切に検索することが重要な課題である。　既存の楽曲検索方法としては楽曲名やアーティスト名などメタデータによる検索が主流である。しかし現在ではその他にも・周囲で流れる音楽・ハミング (鼻歌)・口笛・歌う等のような入力による検索方法が存在する。これらの検索システムは楽曲メタデータがわからなかったり、思い出せなかったりした場合に歌うだけで簡単に直感的な操作で楽曲検索ができることから非常に有用である。このような状況は楽器演奏者にもあるが、歌やハミングをクエリとしたシステムでは楽器音声が想定されていないため検索が望んだ結果を出さない。

本研究では楽器の演奏フレーズをクエリとして、そのクエリに類似したフレーズを検索し得られた類似フレーズを用いて楽曲検索を行うシステムを提案する。また、楽器で楽曲検索が可能であることを示すため和声楽器ではなく基本的に単音でメロディを演奏するエレキベースに着目した。

2　関連研究

2.1　ハミング検索

ハミング検索システムと呼ばれるユーザーの鼻歌をクエリとしてメロディーを検索することができるコンテンツベースの音楽検索手法がある。従来 [1] では音楽信号から FTANet と呼ばれるニューラルネットワークを用いて主旋律の抽出を行いクエリに変換する、そして 3 つの n-gram アルゴリズムと Mode Normalized Frequency を組み合わせた UnifiedAlgorithm により入力フレーズと楽曲データベースとのクエリマッチングを行い類似度が高い楽曲を検索結果としていた。システムの評価実験では 4,400 曲ほどのデータベースを用いて行われ、ベースラインシステムよりも評価指標である MRR,top-n hit rate が上回る結果を得たが、UnifiedAlgorithm を使用した事により検索にかかる処理時間が大きく増加してしまった。そのため、より大規模なデータベースへの応用は困難である。

それに対し本研究では、検索の精度向上に主眼を置き、メロディ抽出の精度が検索結果に影響を与えないよう、約 4.2 万曲からなる大規模な MIDI データセットを使用した。MIDI は各トラックが独立しているため特定のトラックを抽出するのに適しているからである。そしてクエリマッチングにおいてコサイン類似度を用いることで検索にかかる時間を大幅に短縮しつつ、fastText[2] と呼ばれる単語の内部構造を考慮し、高速にテキスト分類や単語埋め込みを生成するニューラルネットワークを利用することで検索の高精度かつ計算の高速化を図る。本研究により、高度な楽曲検索が可能となり、音楽情報処理分野における実用的な応用が期待される。

2.2　melody2vec

melody2vec[3] は、音楽のメロディー情報を分散表現として表現する手法の一つで、word2vec[4] のアルゴリズムをベースにしています。melody2vec では、音符の系列をテキストの単語列に置き換え、その系列から分散表現を学習することで、音楽的な意味や関係性を捉えることができます。楽曲データを前処理し、適切な単位に分割して、それぞれを表すベクトルを生成することで、楽曲の単語ベクトルを作成します。これらの単語ベクトルを用いて、楽曲の特徴を表現することができます。しかし word2vec は学習データセット中に存在しない単語である未知語に対しては対応できません。つまり、学習データセット中に存在しない単語を入力すると、その単語の分散表現を得ることができず、類似したフレーズの検索ができなくなる。

一般的にハミング検索においてもユーザーは誤った入力をする可能性が高いため誤ったフレーズ入力に対応しなければならない。そこで本研究では誤った入力に対応するため FastText を使用した。fastText は、各単語を n-gram に分割して表現し、その n-gram を学習することで単語の分散表現を獲得する。この手法により、未知語でもその n-gram の組み合わせが辞書内に存在する単語と似ている場合、適切な分散表現を生成することができる。つまり、辞書に存在しない単語でも、その構成要素である n-gram を考慮することで適切な分散表現を生成し、類似性を計算することができるため、未知語に対応できる。本研究では 4 つの音符を 1 つのグループとしてテキスト表現化し、これを fastText

---

* **Music retrieval system using monophonic phrase queries played on musical instruments　by Matsushita Riku. (Hosei Univ) et. al.**
[0]Supervisor: Prof. Katunobu Itou

の単語として学習を行った。グループ化は、4 つの音符をまとめて 1 つの単位とし、1 つずつずらしながら実施した。これにより楽曲内のフレーズの漏れを防ぎつつ、楽曲の構造的な特徴を捉えることができるという利点がある。

2.3 fastText

単語の分散表現を学習するモデルとして word2vec の skip-gram モデルをベースとする fastText がある。skip-gram はとある文中の単語からその周辺の単語を予測する教師あり学習モデルである。fastText の skip-gram は文章中のある単語とその単語の n-gram とすることで単語の内部構造を意識し学習ができる。例えば、”where”という単語は 3-gram の場合、”¡wh”,”whe”,”her”,”ere”,”er¿”の 5 つに分けられる。fastText ではこの 5 つの subword 情報と”where”のベクトルを足し合わせることで分散表現を生成する。ここで同じ疑問詞である”when”という単語について考えると subword が 2 つ共通しており”where”と”when”が似ていることを数式的に表現できる。さらに、subword 情報を用いた学習により Word2vec ではできなかった未知語に対しても分散表現を得ることができる。またこのモデルは意味類推や文法理解についても従来の手法より精度が改善されている。

3 提案手法

本研究では、fastText を用いてメロディフレーズの分散表現を事前に学習し、楽曲ベクトル、入力フレーズベクトルの生成に用いた。また、学習のためにメロディのテキスト化とセグメンテーションを行った。セグメンテーションにより切り分けられたフレーズが fastText における単語にあたる。楽曲ベクトルは楽曲の各フレーズ (単語) のベクトルの相加平均によって計算され保存される。fasttext の事前学習にはインターネット上で最大規模の MIDI データセットである Lakh MIDI dataset[5] を使用した。オンライン時には、同様の手法を用いて入力フレーズからベクトルを生成し、保存されている各楽曲ベクトルとの間でコサイン類似度に基づくクエリマッチングを行う。コサイン類似度を利用することで、従来の研究に比べて検索にかかる時間を大幅に削減することが期待される。また、fastText は学習の過程で単語を subword 情報を用いることでユーザーによる誤ったフレーズ入力にも対応可能しつつ、音符の音高や音価の内部情報も考慮することができる。

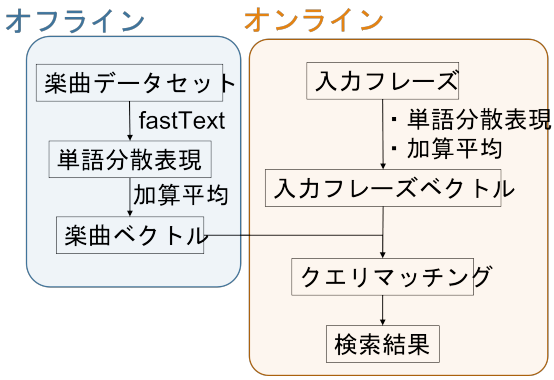

図 1. Overview of this system

3.1 データの準備

学習に用いた Lakh MIDI dataset は、176,581 曲分の MIDI データによって構成されておりメロディ分析を行うことができる最大の楽曲データセットである。MIDI ファイルは各楽器の演奏情報がトラックに分かれている。したがって音源分離のような前処理を必要とせず、ベーストラックを発見できればベースパート抽出が容易に可能であるので大規模な学習データが準備できる。しかし、Lakh MIDI dataset は web 上からクローリングされた MIDI ファイルで構成されているため MIDI ファイル中のどのトラックがベースパートに該当しているのか特定するための規則はない。そこでベーストラック特定のために各トラックの演奏楽器を指定するためのプログラムナンバーを用いた。ベーストラックのプログラムナンバーは”Bass”という単語を含む場合が多い。

本研究ではすべての楽曲のトラック中のプログラムナンバーを小文字化し、その中で”bass”という単語を含む楽曲からベーストラックを抽出した。その結果、全 176,581 曲分のデータの中から 42,829 曲分のベーストラックを抽出することができた。このようにして得られた MIDI データに対し音符情報の書き換えとセグメンテーションによる前処理を行った。

3.2 音符の挿入と書き換え

Lakh MIDI dataset から抽出したベーストラックにおいても不明瞭なメロディをもとの楽譜に近づけるために音長の調整と休符の挿入を行った。

まず MIDI の note はその音が鳴る開始時刻と終了時刻、音高 (pitch) と強さ (velocity) の 4 つの情報を保持している。MIDI には休符の note が存在しないため、本研究では休符を表す note は pitch を 999、velocity を 0 として挿入を行った。休符は 8 分音符以上の無音区間を休符とみなした。8 分音符を挿入することで、不自然な長さの休符が見られなくなった。

また、音価は開始時刻から終了時刻までの時間から単純には求めることができない。それは終了時刻後も余韻として音が鳴っているからである。私の研究では例として開始時刻から終了時刻までが少し 4 分音符分の長さ以上となったら 2 分音符、8 分音符以上なら 4 分音符のように整えた。

3.3 セグメンテーション

melody2vec を学習させるため word2vec における 1 単語を連続する 4 つの音符情報を 1 つのグループとした。

例) 「E1:1/4-E1:1/4-E:1/4-R:1/4」

ここの例は、E1(ミ) の 4 分音符を 3 回鳴らした後に四分休符であるフレーズを表現している。セグメンテーションを抽出されたベーストラック 42,829 曲に対して行った結果、5,180,014 個のフレーズが得られた。これを melody2vec の学習に用いることで各フレーズの分散表現を獲得する。

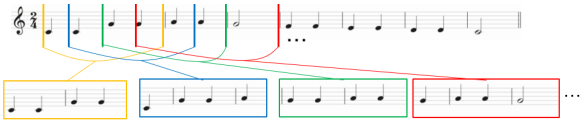

図 2. Segmentation Example

3.4 クエリ生成とフレーズ類似度

楽曲ベクトルは 100 次元の実数値ベクトルで表現される。ベクトルは fastText で獲得した単語分散表現を用い、楽曲のセグメンテーションによる各単語のベクトルを得る。そして本研究ではその加算平均ベクトルを楽曲ベクトルとし保存した。また、入力フレーズに対しても同様な手法でベクトルを獲得し、各楽曲とのクエリマッチングに利用した。

本研究では、入力フレーズと各楽曲ベクトルとのマッチングに、コサイン類似度を採用した。コサイン類似度は、ベクトルの内積とノルムの計算に基づいているため、計算量が比較的少なく、高速な演算が可能である。このため、大規模なデータセットに対しても効率的に類似度を算出することができる。

コサイン類似度は、2 つのベクトルがどれほど似ているかを表す尺度であり、値は 0 から 1 の範囲をとり、1 に近いほど 2 つのベクトルが似ていることになる。具体的には、2 つのベクトル A と B のコサイン類似度は以下の式で定義される。

$$\cos(\mathbf{A},\mathbf{B}) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}$$

ここで、$\mathbf{A}\cdot\mathbf{B}$ はベクトル $\mathbf{A}$ と $\mathbf{B}$ の内積を表し、$|\mathbf{A}|$ と $|\mathbf{B}|$ はそれぞれ $\mathbf{A}$ と $\mathbf{B}$ のノルムを表す。

また、単語の分散表現を用いフレーズ検索した例を以下に示す。例として図 3 の類似フレーズを検索する。

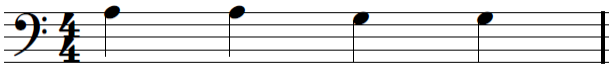

図 3. Example of input phrase

図 3 の類似フレーズとして検索結果上位 5 つのフレーズは上から図 4 から図 8 という結果になった。

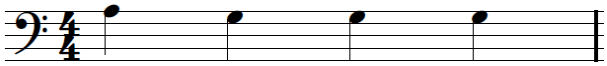

図 4. Similarity #1

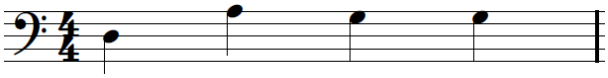

図 5. Similarlity #2

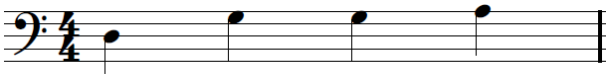

図 6. Similarity #3

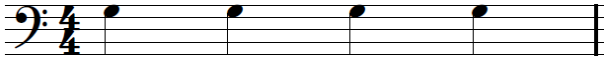

図 7. Similarity #4

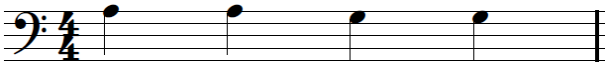

図 8. Similarity #5

これらから類似フレーズとして上位に表れているのは入力フレーズの一つの音符の音の高さが違うだけということが多く実際に類似していることが確認された。

3.5 楽曲のマッチング

本研究では 100 次元の実数値ベクトルで表現された入力フレーズと各楽曲とのコサイン類似度が大きい楽曲を上位にランキングすることで検索結果を決定した。これはコサイン類似度が大きいほど、楽曲中に入力フレーズに近いメロディを含んでいると考えられるからである。

データベース中の楽曲のベクトルは事前に求め保存し、入力フレーズのベクトルはリアルタイムで計算を行う。そして類似度を計算した際、入力フレーズを含む楽曲がランキング上位に現れることが良いマッチングであるといえる。

4　実験

4.1　実験方法

実験には、ランダムに選んだ 200 曲からある 10 から 15 秒程度のフレーズを MIDI から抜き出し、入力フレーズとして検索を行た。また、その検索結果を用いて検索システムの性能を MRR,Top-10 hit rate で評価し従来システムと比較した。MRR、Top-10 hit rate はともに 0 から 1 の範囲の実数値で表され 1 に近いほど目的の楽曲が上位にヒットすることを表す。

$$\mathrm{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\mathrm{rank}_i}$$

4.2　実験結果

実験の結果、MRR は 0.33、Top-10 hit rate は 0.69 と従来システムと同程度の値であり、従来と同等の性能であった。ここで MRR が 0.33 であることは正解楽曲がおよそ上位 3 位までには出現することを表している。また、Top-10 hit rate が 0.69 であることは検索結果上位 10 件に正解楽曲が 69%で存在していることを示す。

	MRR	Top-10 hit rate
Proposed	0.33	0.69
Baseline	0.33	0.73

表 1. MRR and Top-10 hit rate

4.3　考察

提案手法はベースラインシステムと同等の性能を示したが、本システムで利用した約 5 万曲のデータベースの方がはるかに大規模であり、従来研究で使用している約 4,400 曲のデータベースより大規模であり。それにより評価指標は同等であるが本システムは従来より優れた性能を持つシステムであると考える。またクエリ生成の際、単語の分散表現の加算平均をクエリとしていたが、この方法では出現回数の多いフレーズの特徴は反映されやすいが少ないフレーズは逆に反映されにくい。そのためうまくベクトルに反映されていないような部分のフレーズの入力ではうまく検索ができない可能性がある。その他の単語レベルの分散表現から文書レベルの分散表現生成手法についても検討するべきと考える。

5　あとがき

本研究では、MIDI の演奏フレーズを用いた楽曲検索システムを提案した。従来、クエリマッチングにかかっていた計算時間を改善するためにコサイン類似度を用いた。そのため、データベース中の楽曲と入力フレーズをテキスト化し word2vec の一種である Fasttext の事前学習による単語分散表現を用いてベクトルをそれぞれ生成する。検索結果はコサイン類似度が大きい楽曲を上位にランキングし決定した。

実験では評価指標である MRR と Top-10hit rate において従来システムと同等の性能を示したが、本研究では従来の約 10 倍の規模である約 5 万曲の楽曲をデータベースに使用している。そのため、本システムが従来システムより検索システムとして優れていると考えられる。

今後は、実際に楽器を演奏したフレーズを MIDI で録音し、それを用いた楽曲検索の評価を行う。楽器には楽曲の中で主に単旋律で演奏をするエレキベースを選び、MIDI コンバータを用い音声信号を MIDI で録音する。また、本研究では入力フレーズやデータベース中の楽曲のクエリ生成の際、含んでいる単語の分散表現の加算平均をクエリとした。この方法は SWEM と呼ばれる単語分散表現から文書レベルの分散表現を生成する手法の内の、SWEM-aver と同等である。SWEM にはこの他に 3 種類の文書埋め込み生成方法があるため今後は他の文書レベルの埋め込み生成についても行い比較する。

参考文献

[1] Muhammad Ulfi, Rila Mandala, ”Improving Query by Humming System using Frequency-Temporal Attention Network and Partial Query Matching”, 2022 9th International Conference on Advanced Informatics: Concepts, Theory and Applications (ICAICTA).

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov, ”Enriching Word Vectors with Subword Information”, https://doi.org/10.48550/arXiv.1607.04606

[3] Tatsunori Hirai, Shun Sawada, ”Melody2Vec: Distributed Representations of Melodic Phrases based on Melody Segmentation”, Journal of Information Processing Vol.27 278–286 (Mar. 2019)

[4] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. and Dean, J.: Distributed Representations of Words and Phrases and Their Compositionality, Advances in neural information processing systems, pp. 3111–3119 (2013).

[5] https://colinraffel.com/projects/lmd