Blending Two Automatic Playlist Generation Algorithms

James Curbow
Advisor: Matt Anderson
Background

Song Similarity Algorithm + Playlist Generation Algorithm
Background

Song Similarity Algorithm ▼ Expertly Authored Stream Algorithm\(^1\) + Playlist Generation Algorithm ▼ Start-End Algorithm\(^2\)
Research Question

“Will this generate smooth playlists? Can we improve either algorithm?”
Expertly Authored Stream Graphs\(^1\)

<table>
<thead>
<tr>
<th>Playlist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

Similarities

One Way

Two Way
Two Way

Start

A

4/10

B

5/6

C

5/10

D

1/10

D

1/5

F

End
Similarities (example)

One Way

- A to B: 2/3
- B to C: 1/3
- C to A: 2
- A to D: 1
- D to C: 1

Graph 1:
- A connected to B with edge 2
- A connected to C with edge 1
- B connected to C with edge 2

Graph 2:
- A connected to B with edge 2/3
- A connected to C with edge 1/3
- C connected to B

Similarities (example)
Data

• Spotify Web API + SpotiPy
  • ~250,000 Songs
  • ~2,600 Expertly Curated Playlists
Implementation

SpotiPy Text File Scrape

<table>
<thead>
<tr>
<th>Raw_Data_Entry.py</th>
</tr>
</thead>
<tbody>
<tr>
<td>SongLibrary.py</td>
</tr>
</tbody>
</table>

EAS_Graph.py

<table>
<thead>
<tr>
<th>Start_End_Playlist.py</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unbiased_Walk_Playlist.py</td>
</tr>
<tr>
<td>Biased_Walk_Playlist.py</td>
</tr>
</tbody>
</table>

{Both One Way & Two Way Similarities}
The Start-End Song Algorithm

- Picks and orders songs based on increasing Similarity Ratios

**Similarity Ratio:** \( \frac{\text{sim}(S_i, S_e)}{\text{sim}(S_i, S_s)} \)

---

The Start-End Song Algorithm

- Picks and orders songs based on increasing Similarity Ratios

**Similarity Ratio:** \( \frac{\text{sim}(S_i, S_e)}{\text{sim}(S_i, S_s)} \)

---

Start-End Algorithm Results

Issues:

1) Adjacent similarities are never ensured…
2) Every song **needs** to have non-zero similarity to start & end songs…
3) Unclear assumptions about similarity definition…
New Algorithms

- Unbiased Random Walk (URW)
- Biased Random Walk (BRW)
- Greedy Walk (Deterministic)
- Greedy Walk URW
- Greedy BRW

Greediness Parameter

0 → ∞
Similarities

One Way

A -> B: 2/3
A -> C: 1/3
C -> B: 1/3
Similarity Matrices
(Predictions)

(Unbiased Random Walk Algorithm)

<table>
<thead>
<tr>
<th></th>
<th>$S_s$</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_e$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_s$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_e$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Greediness Parameter
Similarity Matrices (Results)
Conclusions

“Will this generate smooth playlists?” No. “Can we improve either algorithm?” Yes.

• How to Fix it?
  • Extend the data
  • Modify algorithm

• Future Work:
  • Evaluative Research
Questions?