libmunin - architecture overview

Input: ①
Music Database
Input: ②
Listen History
③
[Provider]
④
[Weighting]
$D = \begin{pmatrix} d_{11} & \dots & d_{1j} & \dots & d_{1n} \\ \vdots & \ddots & & & \vdots \\ d_{i1} & & d_{ij} & & d_{in} \\ \vdots & & & \ddots & \vdots \\ d_{n1} & \dots & d_{nj} & \dots & d_{nn} \end{pmatrix}$
[Caching] ⑦
Output: ⑧
Recommendations
⑤
S1
S2
S3
S4
0.5
0.8
0.5
0.3
0.2
HistoryBuffer ⑨
⑩ Recommendation Buffer
[Traversing] ⑥
Learning with Time and Input

Inputs

① Music Database:

- The library user feeds songs from the music database (only their metadata).
- The library user also sets an AttributeMask ◦ a set of features most songs have (like a Artist, Title or Genre).
- The internal database can be updated at any time. Also caching it is possible.

② Listen History:

- The library user can feed the last listened songs.
- These can be used to check the Recommendations the library gives.
- This is the main input of learning for libmunin.

Outputs

⑧ Recommendations:

- Giving Recommendation for a Song
- Create dynamic Playlist based on the *Listen History*.
- Ranking of Search Results based on the Similarity of Songs.

Internal:

③ Provider:

- **Song :=** a set of attributes
- **Attribute :=** A feature specific to a Song (e.g. a *Title, Moodbar, ...*)
- **Provider** deliver these attributes (implemented by libmunin)

④ Weighting:

- **Distance** := „*Similarity*" of two songs.
- **DistanceFunction** := Computes a Distance between two Songs.
  The following must apply to a Distancefunction *D*:

  D(i,j) = D(j,i) ∀ i,j ∈ D
  D(i,i) = 1.0 ∀ i ∈ D

- **Attributemask** := A common subset of attributes given by the user of the library, including a weighting for each attribute.

⑤ Distanzmatrix & Graph:

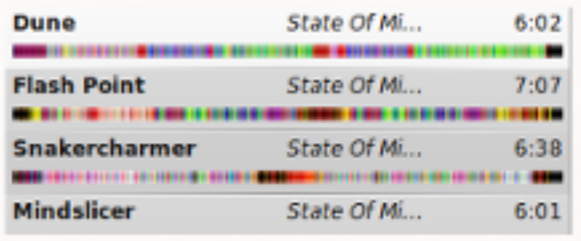
- **DistanceMatrix :=** NxN matrix of D(s1, s2) ∀ s1, s2 ∈ Songs
- Used as Lookup-Table and to build-up the graph.
- **Graph :=** Nodes are Songs; Edges are Distances; Every Song has at most X neighbors.

⑥ Traversing:

- Querying is done by traversing the graph.
- Possible queries:
  - **n**-Similar Songs to **x** (*Breadth-First Search* from **x**)
  - Simlarity of two Song **A** and Song **B** (*Shortest Distance*)
  - Ranking of Search Results (*Similarity with Search-Song* **q**)
- Graph adapts to the user's listening history by modifying edges.

⑦ Caching:

- Hard to calculate Attributes should be stored.
- This includes for example the moodbar:
- Implemented as a SQLite cache usable from the API.



Dune State Of Mi... 6:02
Flash Point State Of Mi... 7:07
Snakercharmer State Of Mi... 6:38
Mindslicer State Of Mi... 6:01

An example of a moodbar

⑨ History Buffer:

- Ringbuffer with **N** Entries at max.
- Holds the latest **N** listened songs.
- Used to evaluate given Recommendations (*Followed* or *Declined*).

⑩ Recommendation Buffer:

- Ringbuffer with **M** Entries at max.
- Holds the latest **M** listened songs.
- Used to *punish* or *reward* songs in the graph.