Lecture 9: Music Recommender Systems

Derek Bridge

Recommending music

- Recommending
  - tracks (or albums or artists or gigs or…)
  - playlists
- But recommenders for playlists
  - typically recommend *sets* of songs
  - but they should be recommending *sequences* of songs
    - in a 'true' playlist, *order* is important

Recommending tracks

- Why we need recommenders for tracks
  - Growth in volume and access
    - unsigned artists; garage bands
    - cheap, mass storage
    - online sales; P2P sharing
    - ubiquitous access (MP3 players, laptops, phones, etc.)
- Primarily, we want to be recommended songs that match our long-term interests for purchase/ download

Recommending a track

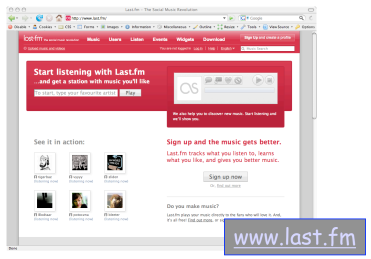
- Collaborative approaches
  - explicit ratings
  - implicit ratings
    - purchasing
    - downloading
    - playing, skipping, …
      (collected by what LastFM.com calls "scrobbling")
    - …
- Problems of knowing exactly which track is being rated
  - due to multiple versions, misspellings, typos, etc.

Recommending a track

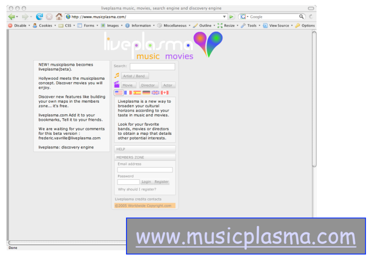
- **Content-based approaches**
  - **Structured descriptions, e.g. ID3 tags in the MP3 format**
    - title, artist, album, genre (from predefined set),…
    - problems
      - incompleteness
      - errors
      - vagueness, subjectivity and incorrect granularity in genre
  - **Unstructured descriptions (sets of keywords), e.g. end-user tags**
    - capturing anything from genre to mood to 'where-seen'
    - Problems
      - unevenness of coverage
      - commonalities hidden by spelling errors, pluralisation, etc.
- **Hybrid approaches are common**

Recommending a track

allmusic.com


www.last.fm

mystrands.com


www.musicplasma.com

Audio analysis for CB approaches

- **Given the problems with structured & unstructured descriptions, why not**
  - **analyze the digital representation of the music itself**
  - **extract features that describe characteristics of the music, e.g.**
    - tempo, rhythm, timbre, instrumentation,…
    - automatic genre classification
- **How?**
  - **experts (Pandora)**
  - **automatically, e.g. Ghanni (?)**
- **Being used in**
  - **music retrieval systems**
  - **recommenders**

www.ghannimusic.com

Paul Lamere, Sun Microsystems

- **A talk given at Recommender Systems 2006 in Bilbao**
  - **The workshop blog (blog.recommenders06.com) contains**
    - a video of his talk
    - his slides
- **Clarification note:**
  - **for Lamere, "content-based" covers audio analysis only**
  - **for us, "content-based" covers audio analysis but other descriptions too (ID3, social tags, etc.)**

Recommending a playlist

- Why we need recommenders for playlists
  - Loss of 'structure' *("MP3 killed the radio star")*
    - the purchasing unit has changed: from album to single song
    - artistic effort (by bands, producers, DJs) to order tracks is being discarded
  - (On the other hand, individuals are creating and sharing playlists)
- When we want listen to tracks from some collection (e.g. ones we own, or ones a radio station can play), we want songs that are
  - based on short-term interests (mood, activity, location, time, etc.)
  - as well as being based on long-term interests
  - and we want 'coherent' sequences

Smart Radio

- C.Hayes (2003): *Smart Radio: Building Community-Based Internet Music Radio,* Ph.D. thesis, TCD
- Used streaming audio
- Simple version
  - Uses collaborative filtering
  - But recommends programmes (playlists), not tracks
    - assumes users have created a large collection of playlists
- More advanced version ("context-boosted collaborative filtering")
  - A hybrid that uses content-based filtering too

Smart Radio's CF

- This is *not* how recommendation works in Smart Radio:
  - Users rate playlists (explicitly with ratings, or implicitly by playing them)
  - Find the nearest neighbours, i.e. users who have similar ratings for playlists
  - Recommend playlists that have been liked by the neighbours
- Why is this unlikely to work?

Smart Radio's CF

- This *is* how it works:
  - Users rate songs (explicitly with ratings or implicitly by playing them)
  - Find nearest neighbours, i.e. users who have similar ratings for songs
  - Score each candidate playlist by summing the following
    - If the active user has rated the song, use his/her rating
    - If the active user has not rated the song, use the average of the neighbours' ratings for that song
  - Recommends the playlists with the highest scores

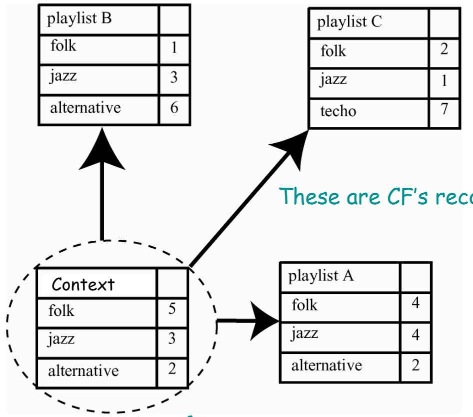
Smart Radio's CF

- Other features
  - Smart Radio does not re-recommend a playlist within a certain period
  - The scoring incorporates a novelty factor to allow users to bias recommendations away from playlists that contain too many songs the user has already rated
  - Users can create new playlists from scratch, or edit ones that have been recommended in order to improve them
- Problems
  - Substantial cold-start problems: needs playlists, and needs song ratings
  - As it stands, Smart Radio is not sensitive to short-term interests (current mood, etc.)

Smart Radio's context-boosted CF

- The last playlist you played is taken to be indicative of the kind of music you'd like to listen to next
- When the user plays a playlist,
  - ID3 info from the tracks in the playlist is gathered and stored
  - Call this the 'context'
- When the user asks for a recommendation for his/her next playlist:
  - The system gets a set of recommendations from CF (as previously)
  - Scores them by how similar they are to the 'context'
  - Recommend those with the highest scores
- What are the problems?

Smart Radio's context-boosted CF



These are CF's recommendations

This is a representation of your previous playlist

Manual playlist creation

- E.g iTunes
  - Standard playlists are created by drag-and-drop
  - Smart Playlists are defined by rules using tags,
    - e.g. Genre is "Pop", Limit to 10 items selected by Random
  - Smart Playlists can even be defined dynamically
    - e.g. PlayCount is greater than 3; e.g. SkipCount is less than 5
  - They suggest you use the Comment tag to enter moods, activities, etc and define Smart Playlists using these
    - e.g. Comment contains "mellow"; e.g. Comment contains "gym"
- Criticisms
  - Huge effort (dragging-and-dropping; defining rules; tagging)
  - Incomplete and vague tags/rules may result in low-quality Smart Playlists
  - These 'playlists' are sets of songs, not sequences of songs
  - Having created them, how do you find the right one to play now?

Mood, activity

- How can a user indicate his/her mood, current activity, etc?
- Explicitly, e.g. http://musicovery.com/
- In the future,
  - biometrics?
- For now,
  - preferred solution is to ask for a *seed* song

Audio analysis for playlists

- As before, we can define a similarity measure on a representation that we compute from audio analysis
- B.Logan & A.Salomon (2001): *A Content-Based Music Similarity Function*, Tech.Report CRL 2001/02, Compaq
  - User chooses a *seed* song
  - System generates a playlist using the songs that are most similar to the seed song
  - But this playlist is a set, not a sequence
- Paul Lamere
  - User chooses two seed songs
  - System generates a playlist connecting the two songs by finding a path thro' a multi-dimensional space
  - This playlist is a sequence, but still based on similarity - users may have different criteria (e.g. contrast)

Reusing existing playlists

- The previous aproaches ignore a valuable resource
- Users contribute playlists to, e.g., MyStrands, LastFM, and thro' iTunes
  - Other sources could be radio programs, web streams, music compilations, DJ sessions
- Presumably, these capture knowledge about which songs 'sound well' in sequence
- We can reuse this knowledge to create new playlists

Reusing existing playlists

- Problems:
  - user-authored playlists are VERY often sets of songs, not sequences, so we should exclude:
    - very short lists
    - very long lists
    - alphabetically-ordered lists
    - …
  - playlitism?
- We look at
  - Claudio Baccigalupo's early Ph.D. work for MyStrands
  - C.Baccigalupo and E.Plaza (2006): Case-based Sequential Ordering of Songs for Playlist Recommendation, Procs. of the 8th European Conference on Case-Based Reasoning

The goal

- Given user’s seed song *s* and desired length *l*, the goal is to find playlist *p* such that
  - *p* contains *s*
  - *p* is of length *l*
  - *p* is varied (does not repeat artist/album or, if it does, then the repetitions are not close)
  - *p* is coherently ordered

Overview

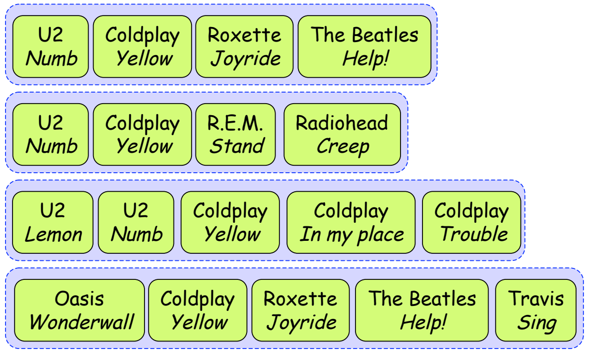
- Offline (in advance), analyse the playlists
  - Find *patterns* (repeats of contiguous songs)
  - Score them (e.g. by frequency)
- Online
  - Asks user for a seed song
  - Retrieve playlists that contain that song
  - Score them (e.g. based on the pattern that occur in them)
  - Take the *k* with the highest scores
  - Combine these *k* playlists

Offline playlist analysis

- Search through playlists for *patterns*
  - Seek sequences of two or more songs that occur with the same order more than once
  - Each pattern is given a pattern score
    - More frequently occurring patterns get a higher score
    - But shorter patterns are penalised
    - And patterns with highly popular songs are penalised
- High frequency sequences are evidence of coherent ordering

Offline playlist analysis

- Here we have
  - One pattern (length 2) that occurs 3 times
  - One pattern (length 3) that occurs 2 times

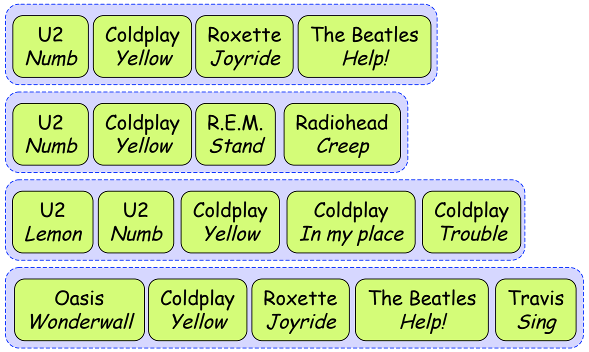


Online playlist retrieval

- **Obtain seed song *s* from user**
- **Consider playlists in the collection that contain *s***
  - **each one of these is given a playlist score, which depends on**
    - Variety
      - Variety of a playlist is initially 1 but the playlist is penalised for every artist that is repeated within n_{artist} songs and every album that is repeated within n_{album} songs, etc.
    - Pattern score
      - Sum up the pattern score for every pattern that occurs in the playlist
  - **Retrieve the *k* playlists that have the highest playlist scores**

Online playlist retrieval

- **Suppose the seed song is U2's *Numb***
  - **How do you think these will score?**

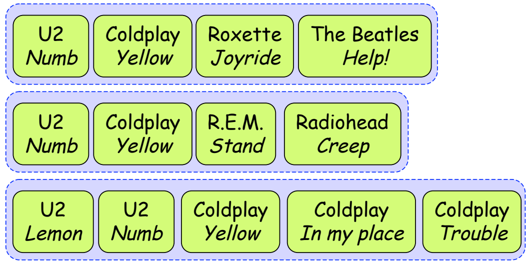


Combining the *k* playlists

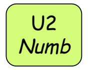
- **We want to use the *k* playlists to produce a new playlist, *p*, of length *l***
- **Here's how:**
  - **Initially *p* contains just *s***
  - **Repeat until *p* is long enough:**
    - For every song *s'* in the *k* playlists, create two candidate extensions of *p:* one in which *s'* is added to the start of *p*; and one in which it is added to the end of *p*
    - Compute the playlist score of each candidate extension
    - Choose the candidate with the highest score: this becomes *p*

Combining the *k* playlists

- **Suppose the seed song is U2's *Numb* and *k = 3***
  - **Retrieved:**



  - **We start with this:**



  - **What are the candidate extensions, and how well to they score?**

Some results

- Try it at MyStrands:
  - labs.mystrands.com/features/cbr/cbr.html
- In some experiments, they used
  - 30,000 MusicStrands playlists
  - *k = 50* (number of retrieved playlists)
  - *l = 10*
  - large values for n_{artist} and n_{album} to discourage repetition

Example playlists

- Input song:
  - American Pie (Don McLean)
- Playlist (with high penalties for popularity):
  - We're An American Band (VV.AA.)
  - Sweet Home Alabama (Lynyrd Skynyrd)
  - More Than a Feeling (Boston)
  - Bad Moon Rising (Creedence Clearwater Revival)
  - American Pie (Don McLean)
  - Mr. Blue Sky (Electric Light Orchestra)
  - Switch (Will Smith)
  - This Love (Maroon 5)
  - Walkie Talkie Man (Steriogram)
  - Walkin' On The Sun (Smash Mouth)

- Input song:
  - American Pie (Don McLean)
- Playlist (with low penalties for popularity):
  - Behind These Hazel Eyes (Kelly Clarkson)
  - Beverly Hills (Weezer)
  - I Just Wanna Live (Good Charlotte)
  - American Idiot (Green Day)
  - American Pie (Don McLean)
  - Hotel California (The Eagles)
  - Cocaine (Eric Clapton)
  - Emerald Eyes (Fleetwood Mac)
  - Carry On Wayward Son (Kansas)
  - Sweet Home Alabama (Lynyrd Skynyrd)

Example playlists

- Input song:
  - Soldier (Destiny's Child)
- Playlist (with high penalties for popularity)
  - Let Me Love You (Mario)
  - Hush (LL Cool J)
  - Red Carpet (Pause, Flash) (R. Kelly)
  - Hot 2 Nite (New Edition)
  - Wonderful (Ja Rule)
  - My Prerogative (Britney Spears)
  - Two Step (Ciara)
  - Soldier (Destiny's Child)
  - Only U (Ashanti)
  - Pass Out (Ludacris)

- Input song:
  - Soldier (Destiny's Child)
- Playlist (with low penalties for popularity):
  - Disco Inferno (50 Cent)
  - Mockingbird (Eminem)
  - Obsession (Frankie J)
  - I Just Wanna Live (Good Charlotte)
  - Boulevard Of Broken Dreams (Green Day)
  - Since U Been Gone (Kelly Clarkson)
  - Two Step (Ciara)
  - Soldier (Destiny's Child)
  - Drop It Like It's Hot (Snoop Dogg)
  - Get Back (Ludacris)

Reflections

- Not highly personalised
  - User's only input is seed song
  - No use of long-term profile of interests
  - No use of feedback
    - Except marginally: if you like a playlist, you could store it (with risk of feedback loops)
- Their latest work considers playlist recommendation in shared listening situations