CmpE 260 - Principles of Programming Languages
Spring 2020 - Project 1

Deadline: 27 April 17:00

Instructor: Fatma Başak Aydemir
Assistants: Alper Ahmetoğlu, Burak çetin
ahmetoglu.alper@gmail.com

1 Introduction

In this project, you will implement a song recommendation system in Prolog. This system will be a rather simple system, yet useful for users to find new songs in their favorite genre with the specified song features.

2 Knowledge Base

You have three main files as your knowledge base: `artists.pl`, `albums.pl`, `tracks.pl`. These are collected with the help of Spotify API. There are three different types of predicates defined as follows:

- `artist(ArtistName, ArtistGenres, AlbumIds).`
  - `ArtistName`: Name of the artist. (string)
  - `ArtistGenres`: List of genres the artist associated with. (list of strings)
  - `AlbumIds`: List of albums of the artist specified with their IDs. (list of strings)
- `album(AlbumId, AlbumName, ArtistNames, TrackIds).`
  - `AlbumId`: A unique ID of an album. (string)
  - `AlbumName`: Name of the album. (string)
  - `ArtistNames`: Names of album's artists. (list of strings)
  - `TrackIds`: List of tracks in the album specified with their IDs. (list of strings)
- `track(TrackId, TrackName, ArtistNames, AlbumName, Features).`
  - `TrackId`: A unique ID of a track. (string)
  - `TrackName`: Name of the track. (string)
  - `ArtistNames`: List of names of track's artists. (list of strings)
  - `AlbumName`: Name of the album. (string)

- Features: [explicit, danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence, tempo, duration_ms, time_signature]

For more information on features, please check Spotify API reference page. In the project, you will only use features with the red color (this is my quite arbitrary choice). All features are numbers.

3 Predicates

In this section, the predicates you are going to implement for your song recommendation system will be explained.

3.1 getArtistTracks(+ArtistName, -TrackIds, -TrackNames) 5 points

This predicate will give us track IDs and track names of an artist.

Examples:

```
?- getArtistTracks("Gorillaz", TrackIds, TrackNames).
   TrackIds = ["7jYUaoOfdcYgUvkK8NnFfx", "7FJ7lHtpbWvPKe3zCoeAWC",
"0IJK096XrKQd14Q6fNqW0e", "5r8c96Zn0ZNHRecVKpFf23",
"0RC23Ua9z3HUD9ssrX2VGj", "6zGTjEZ2zpLbl5Ij7vzTsi",
"4A5FLaZI3Ni5eT0c9fqi8F", "0YHqyIiqJ6QoKKLMDkbcbL",
"5pCCNkuE4nTvEjKl64U7UE"|...],
   TrackNames = ["Tranz", "Kansas", "Sorcererz", "Idaho", "Lake Zurich", "Magic City",
"Fire Flies", "One Percent", "Intro:  I Switched My Robot Off"|...].

?- getArtistTracks("Adele", TrackIds, TrackNames).
   TrackIds = ["4sPmO7WMQUAf45kwMOtONw", "4BHzQ9C00ceJxfG16AlNWb",
"0QLXkoV7vX86QEibgLe6zQ", "7IWkJwX9C0J7tHurTD7ViL",
"7GgQi7JTG4b6J4iEF4RTjF", "4vb4mFvYsr2h6enhjJsq9Y",
"60jkAyuEyOnQKlrppOTYYx", "6oQRvWaoFDJpTTnaWdzhXt",
"5VwSwu1zeiMGAjwe9nUmzZ"|...],
   TrackNames = ["Hello", "Send My Love (To Your New Lover)", "I Miss You", "When
We Were Young", "Remedy", "Water Under the Bridge", "River Lea", "Love in the Dark",
"Million Years Ago"|...].

?- getArtistTracks("Queen", TrackIds, TrackNames).
   TrackIds = ["3YM9o7X3ar7Dm39zXadQSz", "5txoZyuAmtCfmDjUCEphWm",
"34pz8XUZRfw04lk4DPtwa7", "300YN8ebGB90nDuzgz0f30",
"5Q2cQZKPvF4bpcjV3TCTss", "3z8h0TU7ReDPLIbEnYhWZb",
"4ysz5yttr3rpRkvyVMLlRB", "1F9NVicWfNQA5ki8WmEtk8",
"6Am82YdURCRXRvoVUObzzG"|...],
   TrackNames = ["20th Century Fox Fanfare", "Somebody To Love - 2011 Mix", "Keep
Yourself Alive - Live At The Rainbow", "Killer Queen - 2011 Mix", "Fat Bottomed Girls
- Live In Paris", "Bohemian Rhapsody - 2011 Mix", "Now I'm Here - Live At The Hammersmith
Odeon", "Crazy Little Thing Called Love", "Love Of My Life - Live At Rock In Rio"|...].
```

3.2 albumFeatures(+AlbumId, -AlbumFeatures) 5 points

In this predicate, you will return the features of an album. Feature of an album is defined as the average of the features of its tracks. Consider only features that are indicated in red in the previous section.

Examples:

Let's say that we have two tracks in an album with the following features:
`Track1Feats: [0.65, 0.763, 1, 0.134, 0.0108, 0, 0.0782, 0.354]`
`Track2Feats: [0.45, 0.284, 1, 0.0251, 0.814, 1.19e-5, 0.431, 0.337]`
Then the album features should be:
`AlbumFeats: [0.55, 0.5235, 1, 0.07955, 0.4124, 5.95e-6, 0.2546, 0.3455]`
`?- albumFeatures("0cn6MHyx4YuZauaB7Pb66o", AlbumFeatures).` (Mesmerize by SOAD)
`AlbumFeatures = [0.3923, 0.8778, 0.6, 0.08928, 0.09317249999999999, 0.0016232490000000002, 0.22168000000000002, 0.5272].`

`?- albumFeatures("32fmr8WaoHl7XJXnlzyVyX", AlbumFeatures).` (Kuzu Kuzu by Tarkan)
`AlbumFeatures = [0.6772, 0.8321999999999999, 0, 0.07518, 0.0970878, 0.165034, 0.12138000000000002, 0.6681999999999999].`

3.3 artistFeatures(+ArtistName, -ArtistFeatures) 5 points

In this predicate, you will return the features of an artist. Feature of an artist is defined as the average of the features of its tracks. Consider only features that are indicated in red in the previous section.

Examples:

`?- artistFeatures("Tarkan", ArtistFeatures).`
`ArtistFeatures = [0.676012048192771, 0.8034457831325299, 0.3855421686746988, 0.06733493975903615, 0.11073422891566266, 0.04274608807228914, 0.19710722891566274, 0.6231686746987952].`

`?- artistFeatures("Céline Dion", ArtistFeatures).`
`ArtistFeatures = [0.5018247422680412, 0.5428144329896911, 0.5051546391752577, 0.05560721649484537, 0.4388, 0.006481306907216496, 0.27625154639175253, 0.3059134020618557]`

As you see, Tarkan has higher danceability (first feature), energy (second feature), and valence (musical positiveness, last feature) when compared with Céline Dion.

3.4 trackDistance(+TrackId1, +TrackId2, -Score) 5 points

Distance between two tracks depends on the Euclidean distance between their features. Let $x = [x_1, x_2, \ldots, x_8]$, and $y = [y_1, y_2, \ldots, y_8]$ be features of `track 1` and `track 2` respectively. Then, their distance is:

$$D_{track} = \|x - y\|_2 = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_8 - y_8)^2} \quad (1)$$

Lower the distance, the more similar the tracks.

Examples:

```
?- trackDistance("0QZ9l0S8xGFnAiDNHpbNEl", "4jZWeEaLCnwYtLnVEN6BYV", Score).
```
"Tides of Time" "Blue Jeans"

```
   Score = 1.1581405830046458.

?- trackDistance("5EgYSnOYYnIEd74hxuGOTE", "2DcPVRP3xGAX1LxAIJKMWZ", Score).
```
"Kuzu Kuzu" "Afedersin"

```
   Score = 0.22567302278652832.
```

3.5 albumDistance(+AlbumId1, +AlbumId2, -Score) 5 points

Distance between two albums depends on the Euclidean distance between their features, as in `predicate` 3.4. You should first implement `predicate` 3.2 to be able to implement this predicate. When you have the album features, you can calculate the distance as in Equation 1.

Examples:

```
?- albumDistance("49MNmJhZQewjt06rpwp6QR", "0bUTHlWbkSQysoM3VsWldT", Score).
```
"Mezzanine" "Demon Days"

```
   Score = 0.5015809817141805.

?- albumDistance("49MNmJhZQewjt06rpwp6QR", "0lf5ceMub7KQhLfGxCdM06", Score).
```
"Mezzanine" "Death Magnetic"

```
   Score = 0.7323025394358015.
```

3.6 artistDistance(+ArtistName1, +ArtistName2, -Score) 5 points

Distance between two artists depends on the Euclidean distance between their features, as in `predicate` 3.4 and 3.5. You should first implement `predicate` 3.3 to be able to implement this predicate. When you have the artist features, you can calculate the distance as in Equation 1.

Examples:

```
?- artistDistance("Radiohead", "Florence + The Machine", Score).
   Score = 0.4750306486690104.

?- artistDistance("Jennifer Lopez", "Ellie Goulding", Score).
   Score = 0.2927784511950276.

?- artistDistance("Mastodon", "Yalın", Score).
   Score = 0.9529588278452429.
```

3.7 findMostSimilarTracks(+TrackId, -SimilarIds, -SimilarNames) 10 points

Given a track, you will return its 30 closest neighbors (most similar 30 tracks). As you might guess, lower the distance between tracks, the more similar the tracks. In order to implement this predicate, you should implement `predicate` 3.4.

Examples:

```
?- findMostSimilarTracks("1FOernT6bFAm8zscEss4Sm", SimilarIds, SimilarNames).
```
(Omen by The Prodigy)

```
   SimilarIds = ["2PmN7rAGgmPi2JtfVDLCqB", "7lcq0qYpDbjwyEPskseLK7",
"0QZTaRqMg2wLBfPEYWkm4q", "53L6A3I9vf7rgEZnMzx54E",
"2nAGcE9BI0vhSHyM2AEGNT", "1NvioR5fwZCNgcwnc3IH4j",
"38QNpJKe98IrHkD2iZ73xu", "28AgTOIZke8nitD0NNeb3G",
"282tIw1ajuDfOZPewFWPwP"|...],
   SimilarNames = ["Restless - Extended Mix", "Don't Stop At The Top", "Captain Dread
- Zexos Free Troupe Mix", "Mountain at My Gates", "Leitbild", "Has It Come to This?
- Zed Bias Dub Mix", "Mine", "Up All Night", "Where Were You When The Lights Went
Out - Hirshee Remix"|...].
```

```
?- findMostSimilarTracks("7f9I5WdyXm5q1XqnSYgQZb", SimilarIds, SimilarNames).
```
(Brianstorm by Arctic Monkeys)

```
   SimilarIds = ["6807DEj6vL5hhuWU3Qgaw9", "2QKfks5yuzxNtGj16TVL8y",
"3R2ujuKBNOLLmI3BN4pQYh", "4hn85MteYZcc5WMVPYT2Ua",
"1qYDzLpv3mUviEIYN5Q4vG", "2iR5ItVdY6WTMvPEgNT3ri",
"15vTJnpJ9ypbisykXVaawK", "0CHZjALtuGGST7sOEqofJM",
"6V68ItawQkQlZhYIf1S86C"|...],
   SimilarNames = ["Going My Way!", "Grammatizator", "Built To Survive", "Milestone",
"An Evening of Extraordinary Circumstance", "Are You There Margaret?  It's Me, God",
"Know", "FEED THE FIRE", "Crazy = Genius"|...].
```

3.8 findMostSimilarAlbums(+AlbumId, -SimilarIds, -SimilarNames) 10 points

Given an album, you will return its 30 closest neighbors (most similar 30 albums). In order to implement this predicate, you should implement `predicate` 3.5.

Examples:

```
?- findMostSimilarAlbums("1Mu6HgmdfdiQTOZ8mPEyFU", SimilarIds, SimilarNames).
```
(Icky Thump by The White Stripes)

```
   SimilarIds = ["7oadUYM1zfPFjZYLavQkjL", "30ly6F6Xl0TKmyBCU50Khv",
"68VSkHK4UvupiUB0g6Dbno", "2xUECIo0BalLsOl686J9kt",
"1RuYprt6Qlqu286h1f4dzZ", "55ClRfRsxY2bB3E72MekaC",
"3gk7wQvzdHKKLKkBWpbSV9", "0DvD10MTfOLbpe3gu61y1P",
"0UG3t2Ki2dy4u6c75HqH64"|...],
   SimilarNames = ["Head Carrier", "The Colour And The Shape", "First Taste", "Elephant
Stone", "Binaural", "Four Minute Mile", "Fisheye", "A Cabinet Of Curiosities", "War
of Kings (Special Edition)"|...].
```

```
?- findMostSimilarAlbums("32fmr8WaoHl7XJXnlzyVyX", SimilarIds, SimilarNames). (Kuzu
Kuzu by Tarkan)
   SimilarIds = ["0yoy3Hh3yFZGb2KdXw10GQ", "4rVINT2gMVCFNjekXhBnAY",
"40BygoJE7fxxXls20NtptS", "51vRvV83RdWGP9FpzGe4SQ",
"2eVhDmqxQ9nzSs34hhXBFR", "6gCIerPZhK4Suewkg3NjOZ",
"4uzBMJpPFHst5PiCduRKFH", "4VxqrQsfvo0e9GBqcH1qb0",
"1oXQHuGT3v1BxM0gYVcWTg"|...],
   SimilarNames = ["Metamorfoz Remixes", "Killing Me Softly", "Yaqti", "The Time
Is Now (Deluxe)", "Spilling Over Every Side", "私を鬼ヶ島に連れてって", "Setareh",
"Arash", "Kusursuz 19"|...].
```

3.9 findMostSimilarArtists(+ArtistName, -SimilarArtists) 10 points

Given an artist, you will return its 30 closest neighbors (most similar 30 artists). In order to implement this predicate, you should implement `predicate` 3.6.

Examples:

```
?- findMostSimilarArtists("Halil Sezai", SimilarArtists).
   SimilarArtists = ["H.E.R.", "Ebi", "London Grammar", "Lana Del Rey", "Andreya
Triana", "Mohsen Namjoo", "Tinashe", "Ekamatra", "Nana Caymmi"|...].

?- findMostSimilarArtists("Avicii", SimilarArtists).
   SimilarArtists = ["Olly Murs", "Jennifer Lopez", "Katy Perry", "Sia", "Dua Lipa",
"P!nk", "Kesha", "Red Hot Chili Peppers", "The Knocks"|...].

?- findMostSimilarArtists("deadmau5", SimilarArtists).
   SimilarArtists = ["Bola", "Pantha Du Prince", "Plaid", "Hudson Mohawke", "Tycho",
"Massive Attack", "Sascha Braemer", "The Album Leaf", "Caribou"|...].
```

3.10 filterExplicitTracks(+TrackList, -FilteredTracks) 5 points

In this predicate, you will filter tracks which has explicit lyrics and return the filtered tracks.

Examples:

```
?- TrackList = ["0gvQoTWRxsW5Rd7KgPp0u3", "6HZILIRieu8S0iqY8kIKhj", "3B0irDyS69y5eAz15xV2Ee"],
filterExplicitTracks(TrackList, FilteredTracks).
   FilteredTracks = ["3B0irDyS69y5eAz15xV2Ee"].
   Some Kendrick Lamar songs are filtered.
```

3.11 getTrackGenre(+TrackId, -Genres) 5 points

In this predicate, you will return genres of a track. Genres of a track is a list of genres that its artists associated with. There may be a multiple of artists of a track. In this case, genres are the concatenated list genres of both artists. If there is no genre associated with the artist, return an empty list.

Examples:

```
?- getTrackGenre("0QZ9l0S8xGFnAiDNHpbNEl", Genres).
   Genres = ["dutch metal", "gothic metal", "gothic symphonic metal", "power metal",
"progressive metal", "symphonic metal"].
```
(Tides of Time by Epica)

```
?- getTrackGenre("0gvQoTWRxsW5Rd7KgPp0u3", Genres).
   Genres = ["conscious hip hop", "hip hop", "rap", "west coast rap"].
```
(HUMBLE. by Kendrick Lamar)

3.12 discoverPlaylist(+LikedGenres, +DislikedGenres, +Features, +FileName, -Playlist) 30 points

The ultimate predicate. In this predicate, the user will enter a list of liked genres, a list of disliked genres, features (danceability, energy, ...) and the recommendation system will return 30 tracks with respect to these settings. The genre of each track in the playlist should include at least one string from `LikedGenres`, and no string from `DislikedGenres`. However, the user might not be very informed about genres. Therefore, the system should give results in a more general fashion. For example, the track "Tides of Time" in the examples of `predicate` 3.11 should be included in the playlist if the user sets `LikedGenres = ["metal"]`, even if `"metal"` is not in the genre list by itself, but a substring of one or more of the genres. If the user enters `LikedGenres = ["pop"]`, every track which has a substring `"pop"` in at least one of its genres is a valid track. Likewise, if the user does not want to listen to jazz music, she/he does not have to know every specific jazz genre. She/he enters `DislikedGenres = ["jazz"]` and there will not be any track with genres including `jazz` as a substring.

The playlist should be sorted with respect to distances between tracks and `Features`. Tracks with smaller distances (more similar to the `Features`) should appear first.

The playlist should be written to the file with name `FileName` in the following order:

```
[trackid1,trackid2,trackid3,...,trackid30]
[trackname1,trackname2,trackname3,...,trackname30]
[artists1,artist2,artists3,...,artists30]
[distance1,distance2,distance3,...,distance30]
```

Example outputs will be provided with its query.

4 Documentation

Please explain what each predicate is for with comments in the code. Codes with no comments will lose points.

5 Submission

You will submit two files: `solution.pl`, `feedback.txt`. Your code should be in one file named `solution.pl`. First four lines of your `solution.pl` file must have exactly the lines below since it will be used for compiling and testing your code automatically:

```
% name surname
% student id
% compiling: yes
% complete: yes
```

The third line denotes whether your code compiles correctly, and the fourth line denotes whether you completed all of the project, which must be `no` if you're doing a partial submission. This whole part must be lowercase and include only the English alphabet. Example:

```
% alper ahmetoglu
% 2012400147
% compiling: yes
% complete: yes
```

I am interested in your feedback about the project. In the `feedback.txt` file, please write your feedback. This is optional, you may leave it empty and omit its submission.

6 Tips and Tricks

Although the project seems long at first glance, it can be done in a reasonable amount of time. Therefore do not panic.

- Do not rush. First think about the requirements, what you need, how you can solve it in a modular way.
- Try to formalize the problem, then try to convert the logic formulate to Prolog.
- You can use findall/3, bagof/3 or setof/3.
- You can use extra predicates and it is highly recommended. The ones given above are compulsory.
- If a predicate becomes too complex, either divide it into some predicates or take another approach. Use debugging (through trace/1), approach your program systematically.

For those who want to test their solution on the full data.
It takes 1-2 mins to load the data on my system.

https://drive.google.com/file/d/1jfJaoXATHe2xZod3qUAo0APtEJGmicOO/view?usp=sharing
https://drive.google.com/file/d/1DFDt01ZZ2RZ_8Rhgo4q_oOYxcjJ-GgQV/view?usp=sharing
https://drive.google.com/file/d/1MAgYRRu5cI40J1RpX3iE1A0RUyQxflIX/view?usp=sharing