La Recherche d'Images Similaires
Projet intégrateur

Partie #3: Logique (Prolog)

uOttawa

CSI2520[A] Paradigmes de programmation

Présenté pour Robert Laganière

Yassine Moumine 300140139

Date: le 08 Avril 2024

Table de Matiere

Table de Figures

Introduction

Dans le monde moderne, la prolifération rapide des images nécessite des outils informatiques avancés pour analyser, rechercher, classer et découvrir des images d'intérêt. Ce projet vise à implémenter une méthode de recherche d'images similaires en utilisant la **programmation logique en Prolog**.

Données d'Entrée

La base de données d'images est fournie au format JPG, avec les histogrammes précalculés enregistrés dans des fichiers texte. En outre, 16 images de requête sont fournies au format JPG et PPM. Pour cette partie nous utilisons **le format texte** qui contient les histogrammes des images précalculées.

Objectifs:

- Résoudre le problème de similarité en décodant les histogrammes des images.
- Utiliser la programmation logique pour définir les prédicats manquants.

Solution Algorithmique

1. Afin de faciliter l'exécution du programme et pouvoir tester toutes les images requises dans un seul dossier. J'ai ajouté le prédicat **'read_query/2'**, qui construit le chemin complet vers le fichier d'histogramme de l'image de requête, en utilisant le chemin prédéfini avec **'query(DirectoryName)'**, et puis lit l'histogramme.

```
% query(DirectoryName)
% this is where query histogram files are located
query('.\\queryImages\\').

% read_query(Filename,ListOfNumbers)
% get the full path for each query image and read its histogram
read_query(Filename, Histo):- query(P),atom_concat( P, Filename,
FullPath),read_hist_file(FullPath, Histo).
```

Figure 1: Code du predicat 'read_query/2'

2. Le prédicat **'compare_histograms/4'** compare l'histogramme de l'image de requête avec les histogrammes de toutes les images dans le

dataset. Pour chaque fichier de l'histogramme de la base de données, il construit le chemin complet vers ce fichier, lit l'histogramme, puis calcule le score de similarité entre l'histogramme de la requête et l'histogramme du fichier de base de données en utilisant le prédicat de calcul d'intersection de l'histogramme suivant.

```
% compare_histograms(QueryHisto,DatasetDirectory,DatasetFiles,Scores)
% compares a query histogram with a list of histogram files
compare_histograms(QueryHisto, DatasetDirectory, DatasetFiles, Scores) :-
    findall((Filename, S),
            (member(Filename, DatasetFiles),
             atom_concat(DatasetDirectory, Filename, FullPath), % Get the full
path to dataset images
             read_hist_file(FullPath, DatasetHisto), % Read the dataset histogram
             histogram_intersection(QueryHisto, DatasetHisto, S)), % Calculate
the intersection
            Scores).
```

Figure 2: Code du predicat 'compare_histograms/4'

3. Ce prédicat **'histogram_intersection/3'** est responsable de calculer l'intersection entre deux histogrammes et retourne leur valeur de similarité entre 0.0 et 1.0.

```
% histogram_intersection(Histogram1, Histogram2, Score)
% compute the intersection similarity score between two histograms
% Score is between 0.0 and 1.0 (1.0 for identical histograms)
histogram_intersection(H1, H2, S) :-
    maplist(d, H1, H2, MinPairs),
    sumlist(MinPairs, SumMinPairs),
    sumlist(H1, SumH1),
    S is SumMinPairs / SumH1.
d(X, Y, Min) :- Min is min(X, Y).
```

Figure 3: Code du predicat 'histogram_intersection/3'

Résultat Attendu

1. L'objectif est de trouver les 5 images les plus similaires à chaque image de requête, en utilisant l'intersection de leurs histogrammes comme vu précédemment dans la partie object-oriented et concurrence.

```
?- similarity_search('q00.jpg.txt',S).
S = [('1144.jpg.txt', 1), ('3806.jpg.txt', 0.7074537037037038), ('3714.jpg.txt', 0.668431712962963), ('3756.jpg.txt
', 0.6620254629629629), ('2462.jpg.txt', 0.6491030092592592)] .

?- similarity_search('q14.jpg.txt',S).
S = [('2193.jpg.txt', 0.4463136574074074), ('3108.jpg.txt', 0.38057291666666665), ('3318.jpg.txt', 0.37972800925925
926), ('4028.jpg.txt', 0.3712037037037037), ('3085.jpg.txt', 0.3705439814814815)] .

?- similarity_search('q01.jpg.txt',S).
S = [('3588.jpg.txt', 0.5703645833333333), ('2536.jpg.txt', 0.561255787037037), ('3553.jpg.txt', 0.5365625), ('1875
.jpg.txt', 0.5282986111111111), ('2592.jpg.txt', 0.5049710648148148)] .

?- similarity_search('q04.jpg.txt',S).
S = [('1799.jpg.txt', 0.8465046296296296), ('848.jpg.txt', 0.4933969907407407), ('4261.jpg.txt', 0.4890682870370370
3), ('1800.jpg.txt', 0.4714351851851852), ('1937.jpg.txt', 0.4680844907407407)]
```

Figure 4: Résultats de sortie

Références

[1] prolog - How to concatenate two atoms/strings? (n.d.). Stack Overflow. https://stackoverflow.com/questions/16898156/how-to-concatenate-two-atoms-strings

[2] SWI-Prolog -- member/2. www.swi-prolog.org. https://www.swi-prolog.org/pldoc/man?predicate=member/2.

[3] Trying to understand maplist. https://www.reddit.com/r/prolog/comments/r14bry/trying_to_understand_maplist/.

[4] Amzi! inc. Adventure in Prolog tutorial. www.amzi.com. https://www.amzi.com/AdventureInProlog/a1start.php.