CSE3018 : Content Based Image and Video Retrieval

Lab - 11

Naga Harshith Bezawada
19BCE1547

CODE

```
% The main handler that reads the csv, calculates all the
distances using
% the different distance measure from the query record. After,
finding the
% individual distance, the individual records are ranked and the
top 20
% records for each metric are recorded

records = csvread("./datas/data.csv");
query = [4, 20, 70, -1, 200, 0.0025, 45234, 15.23, 19.45, 490];


% Initializing vectors to store the distance of each record
sum_absolute_differences = [];
sum_squared_absolute_differences = [];
euclidean_distances = [];
city_block_distances = [];
canberra_distances = [];
maximum_value_distances = [];
minkowski_distances = [];
```

```
chi_square_distances = [];

hamming_distances = [];

cosine_distances = [];

earth_movers_distances = [];

pearson_correlation_coefficients = [];



% Calculating the distance of each record from the query and
storing them

% in the respective list

for i=1:length(records)

     sum_absolute_differences = [sum_absolute_differences,
sum_absolute_difference(records(i, :), query)];

     sum_squared_absolute_differences =
[sum_squared_absolute_differences,
sum_squared_absolute_difference(records(i, :), query)];

     euclidean_distances = [euclidean_distances,
euclidean_distance(records(i, :), query)];

     city_block_distances = [city_block_distances,
city_block_distance(records(i, :), query)];

     canberra_distances = [canberra_distances,
canberra_distance(records(i, :), query)];

     maximum_value_distances = [maximum_value_distances,
maximum_value_distance(records(i, :), query)];

     minkowski_distances = [minkowski_distances,
minkowski_distance(records(i, :), query)];
```

```
     chi_square_distances = [chi_square_distances,
chi_square_distance(records(i, :), query)];
     hamming_distances = [hamming_distances,
hamming_distance(records(i, :), query)];
     cosine_distances = [cosine_distances,
cosine_distance(records(i, :), query)];
     earth_movers_distances = [earth_movers_distances,
earth_movers_distance(records(i, :), query)];
     pearson_correlation_coefficients =
[pearson_correlation_coefficients,
pearson_correlation_coefficient(records(i, :), query)];
end


% Declaring the column names
column_names = {};
for i=1:10
     column_names{end+1} = sprintf('%s%d', 'feature', i);
end
column_names{end+1} = 'sum_absolute_differences';
column_names{end+1} = 'sum_squared_absolute_differences';
column_names{end+1} = 'euclidean_distance';
column_names{end+1} = 'city_block_distance';
column_names{end+1} = 'canberra_distance';
column_names{end+1} = 'maximum_value_distance';
column_names{end+1} = 'minkowski_distance';
```

```
column_names{end+1} = 'chi_square_distance';
column_names{end+1} = 'hamming_distance';
column_names{end+1} = 'cosine_distance';
column_names{end+1} = 'earth_movers_distance';
column_names{end+1} = 'pearson_correlation_coefficient';

% Table for storing the data
info_table = cell2table(cell(0, size(column_names,2)),
'VariableNames', column_names);

% Creating the final dataframe with the records and
corresponding distance
% for all the distance metrics
final_table = [...
     records ...
     sum_absolute_differences' ...
     sum_squared_absolute_differences' ...
     euclidean_distances' ...
     city_block_distances' ...
     canberra_distances' ...
     maximum_value_distances' ...
     minkowski_distances' ...
     chi_square_distances' ...
     hamming_distances' ...
     cosine_distances' ...
     earth_movers_distances' ...
```

```
    pearson_correlation_coefficients'];

for i=1:length(final_table)
    new_row = {};
    for j=1:length(final_table(i, :))
    new_row{end+1} = final_table(i, j);
    end
    info_table = [info_table; new_row];
end

% Storing the complete data with calculated distances in xls
file
writetable(info_table, './datas/data_with_distances.xls');



% Finding the rank of each record according to the individual
distance
% metrics
% Table for storing the rank of each record for the
corresponding distance
% metric
rank_table = cell2table(cell(0, size(column_names(11:end),2)),
'VariableNames', column_names(11:end));

% Calculating the rank of each record
ranks = [];
```

```
for i=1:length(column_names(11:end))
     [data, idx] = sort(final_table(:, i));
     ranks = [ranks idx];
end
for i=1:length(ranks)
     new_row = {};
     for j=1:length(ranks(i, :))
     new_row{end+1} = ranks(i, j);
     end
     rank_table = [rank_table; new_row];
end

% Storing the complete data with calculated distances in xls
file
writetable(rank_table, './datas/distance_ranks.xls');



% Finding the top 20 indices in ascending order for each
distance metric
% Table for storing the rank of each record for the
corresponding distance
% metric
top_table = cell2table(cell(0, size(column_names(11:end),2)),
'VariableNames', column_names(11:end));

top = [];
```

```
for i=1:size(ranks, 2)
   idx = find(ranks(:, i) < 21);
   top = [top idx];
end
for i=1:length(top)
     new_row = {};
     for j=1:length(top(i, :))
     new_row{end+1} = top(i, j);
     end
     top_table = [top_table; new_row];
end

% Storing the complete data with calculated distances in xls
file
writetable(top_table, './datas/top20_records.xls');
```

Euclidean_distance

```
function[x] = euclidean_distance(record, query)
% Function to find the euclidean distance between the record and
the query
x = norm(record - query);
return
```

Sum_squared_absolute_difference

```
function[x] = sum_squared_absolute_difference(record, query)
```

```
% Function to find the sum of absolute squared difference between the record and
% the query
x = sum((record - query).^2);
return
```

Sum_absolute_difference

```
function[x] = sum_absolute_difference(record, query)
% Function to find the sum of absolute difference between the record and
% the query
x = sum(abs(record - query));
return
```

Pearson_correlation_coefficient

```
function[x] = pearson_correlation_coefficient(record, query)
% Function to find the pearson correlation coefficient between the record
% and the query
x = corr2(record, query);
return
```

Minkowski_distance

```
function[x] = minkowski_distance(record, query)
```

```
% Function to find the minkowski distance between the record and the query

x = pdist2(record, query, 'minkowski');

return
```

Maximum_value_distance

```
function[x] = maximum_value_distance(record, query)

% Function to find the maximum value distance between the record and the query

x = max(abs(record - query));

return
```

Hamming_distance

```
function[x] = hamming_distance(record, query)

% Function to find the hamming distance between the record and the query

x = pdist2(record, query, 'hamming');

return
```

Earth_movers_distance

```
function[x] = earth_movers_distance(record, query)

% Function to find the earth movers distance between the record and the

% query
```

```
x = sum(abs(cumsum(record) - cumsum(query)));

return
```

Cosine_distance

```
function[x] = cosine_distance(record, query)

% Function to find the cosine distance between the record and the
query

x = 1 - (sum(record .* query) /
sqrt(sum(record.^2)*sum(query.^2)));

return
```

City_block_distance

```
function[x] = city_block_distance(record, query)

% Function to find the city block distance between the record and
the query

x = sum(abs(record - query));

return
```

Canberra_distance

```
function[x] = canberra_distance(record, query)

% Function to find the canberra distance between the record and the query

x = (abs(record - query)) / (abs(record) + abs(query));

return
```

Chi_square_distance

```
function[x] = chi_square_distance(record, query)

% Function to find the chi square distance between the record and the query

x = sum((record - query).^2 / (record + query)) / 2;

return
```