Project Algorithms & Data Structures

Group 35

Ward de Muer & Garben Tanghe

**Content Based Filtering**

**Scenario 0:** A HashMap is useful because it provides key-value access to data.

Other options are stacks, queues and dictionaries.

HashMaps:

Advantages of using HashMaps are:

Disadvantages of using HashMaps are:

Stacks:

Advantages of using stacks are: easy to store.

Disadvantages of using stacks are: you are restricted to the last added value, which should be removed before you can access the second last value.

Queues:

Advantages of using queues are: easy to store.

Disadvantages of using queues are: you are restricted to the first added value, which should be removed before you can access the second value.

Dictionaries:

Advantages of using dictionaries are:

Disadvantages of using dictionaries are:

**Scenario 1:** - The complexity of Dynamic1 is O(n5) in worst case.

- A worst case scenario is that all the letters in String s are the same. There will be may (equal) SquareSubsequences. The longest SquareSubsequence will have a length of n when n is even or n-1 if n is uneven, with n = s.length();

- The complexity of Dynamic2: not implemented

**Collaborative Filtering**

**Scenario 0:** Idem as scenario 0 in ContentBasedFiltering.

**Scenario 2:** The cosine distance is totally wrong when, 0 is a possible rating.

Let’s consider these 2 cases:

Case 1:

|  |  |  |
| --- | --- | --- |
|  | Movie A | Movie B |
| User A | 0 | 5 |
| User B | 0 | 5 |
| User C | 0 | 5 |
| User D | 0 | 5 |
| d(A, B) = 1 - = 1 - = 1 | | |

Case 2:

|  |  |  |
| --- | --- | --- |
|  | Movie A | Movie B |
| User A | 0 | 0 |
| User B | 1 | 0 |
| User C | 0 | 0 |
| User D | 0 | 1 |
| d(A, B) = 1 - = 1 - = 1 | | |

Attention: at least 1 rating should be non-zero!

In case 1, the distance could be much bigger and we do get the biggest distance.

In case 2, the distance is not big at all, but we get an equally big distance as in case 1.

**Scenario 3:** When a movie is not rated, there is a problem. We can solve it by giving that movie a rating of 2.5. If a user hast just watched one of the greatest movies he’s ever seen or one of a kind he never wants to witness again, he’s much more likely to give this movie a rating. Whereas if the user thinks the movie is rather mediocre, he might not have the needs to express his / her feelings about the movie. Giving this movie an average rating of 2.5 might be a good solution to get rid of movies which don’t have a rating.

**Scenario 4:** A solution for giving a movie a rating for each user is to give it a weighted average, based on other users who have seen the movie. The smaller the distance between 2 users, the bigger the weight and the other way around.

Another method is to use content based filtering. When a user didn’t rate a movie, find the closest movie that the user did rate and give it the same rating. When 2 movies are very similar, the same user should give it almost the same rating.

A similar method to this one, without using content based filtering is to just look at some of the similar movies to a specific movie, that the user did rate and calculate an average rating for it.

Working further from the last method, we could recommend only the movies with the highest rating by saving the newly created ratings in a fixed size priority queue.