# Week 2 - Tasks 3&4 weekly report

Rémi Pichon - 15 february 2016

## What has been implemented

### Task 3

I implemented a dynamic weight calculation for genres and directors for each user. This computation has been successfully implemented for each user on demand (it is not pre-computed in the Reader) in the Recommender class. I use recall over precision and topN over precision to asset my result.

To complete task 3, several class have been added an other updated.

- Execute (created)

MaxRecommender.java has beed added inspired by MeanRecommender.java. It perform a **max commendation** instead of a mean.

- CaseSimilarity (updated)

add method `public double getSimilarity(FeaturesWeight featuresWeight, Case c1, Case c2);` in order to give dynamic weight to similarity.

- RecommenderNonPersonalised and RecommenderPersonalised (created)

new interfaces in order to have a custom Recommender for either personalised and non personalised. They both get a common interface Recommender. Basically RecommenderNonPersonalised is the old Recommender.

- MaxRecommenderPersonalised

new Recommender that improve *getRecommendations* to compute dynamic genres and directors weight for each user. This method is called once for every user, there is no overhead of doing the dynamic weight computing here instead of in the reader. `W = 1-#distinct /(#profilemovies)` is used to compute both dynamic weight.

- ExecuteTaskThree (created)
- FeaturesWeight (created)

model to exchange dynamic weights

### Task 4

I implement a random selection over a list 10 times larger than the asked topN. Even with a random selection diversity don't improve successfully . Even with a random algorithm the diversity should be better (and the recall over precision should drop).

To complete task 4, several class have been added an other updated.

- ExecuteTaskFour (created)

Method *evaluateAndPrintResultForDiversity* has been implemented in order to output recommendations diversity

- Execute (updated)

`evaluateAndPrintResultForDiversity` to display diversity

- Evaluator (updated)

add method `public double getDiversity(int topN)` to compute diversity for a topN bounded list.
add `private ArrayList<Integer> getIntersection(Integer userId, final ArrayList<Integer> recommendations, final Map<Integer,Double> testProfile, final int topN)`. This method perform a random diversity.

## Running instruction

There is one class ExecuteTaskThree.java with a main to run the implemented personalised case similarity feature weights for both the director and genre features.
There is one class ExecuteTaskFour.java with a main to run the random diversity enhance algortythm.