Extended Matching Documentation

# 1. Feature description

Extended Matching feature is designed to estimate cross similarity of several persons by comparing user’s answers for specific questions (extended questions). Each user with active account can answer these questions by choosing appropriate variants (from 0 to all of given). In case if user answers to extended question, he should also choose question importance from predefined (this value should describe how important is the question for user).

Each extended question has predefined answers. Every answer is comparable to all other answers by numeric value (answer similarity). The maximum value of answer similarity is the value of comparing answer to itself. All these values should be predefined. There are also 4 predefined variants of question importance with corresponding numeric value.

The result of estimating cross similarity between 2 users by Extended Matching is called Matching Score. Matching Score is numeric value in range from predefined basic up to 100. Algorithm of it’s calculation is described below.

# 2. High Level Design

**2.1. User Matching Score calculation formula.**

In order to calculate User Matching Score between 2 users following data is needed:

* user’s answers for Extended Questions;
* denoted Question Importance for each answered Extended Question;

To simplify description, user with active account who requested the matching score is called user origin, and the user who is being matched against is called matching user.

where:

n – number of answered Extended Questions by user origin;

m – number of answered Extended Questions by matching user;

BasicScore – predefined value which indicates minimum possible User Matching Score. This value I set in CoreMatchingService and by default set to 30;

Qiorigini – value of Question Importance chosen by user origin for the ith Extended Question;

Qimatchi – value of Question Importance chosen by matching user for the ith Extended Question;

AnsOrigini – answers chosen by user origin for the ith Extended Question;

AnsTomatchi – answers chosen by matching user for the ith Extended Question;

AnSim(X, Y) – predefined value which indicates how similar are answer X to answer Y for the same Extended Question.

**2.2. User’s answers saving.**

After user finished answering to Extended Questions, his results are sent to UserQuestionExtendedAnswerService, where service calculates max score for given Extended Question answers as describe in formula (MaxScoreOrigin). Also, all previous Matching Score values are deleted from database where current user is present as user origin or matching user.

**2.3. Getting User Matching Scores.**

In order to get all User Matching Scores between specific user and all other users, who are found to be matched by General Matching Feature, UserMatchingScoreService is designed. It’s method getUserMatchingScores takes as input User object and returns collection of pairs in form “userId” and “matchingScore” for each user found to be matched by General Matching Feature. “matchingScore” is retrieved from database, but in case if there is no stored value, it is calculated by CoreMatchingService and after is saved to database.

**2.4. User Matching Score calculation.**

Basic formula described in p.2.1 is implemented in CoreMatchingService in method calculateMatchingScore. This method retrieves data about maximum scores for user origin and matching user from database along with corresponding answers for Extended Questions by both users.

For each question CoreMatchingService gets value of answers similarity from database. In case if one of the users has multiple answers for the same Extended Question, maximum value of answer similarity is taken. For example, if user origin answers for Extended Question with variant “a”, and matching user answers for the same Extended Question with variants “b” and “c”, and answers similarity values are set as shown in table below, the value “3” (between variants “a” and “c”) will be taken.

|  |  |  |  |
| --- | --- | --- | --- |
|  | a | b | c |
| a | 5 | 1 | 3 |
| b | 1 | 5 | 4 |
| c | 3 | 4 | 5 |

# 3. Extended Matching feature implementation

**3.1. Controllers.**

There is only one controller – UserMatchingScoreController. This controller contains following mappings:

* GET: /matching/extended/
* POST: /matching/extended/

First mapping is mapped to getUserMatchingScores() method. This method takes as input authenticated user and returns collection of pairs “userId”: “matchingScore” as output. This method is designed to return User Matching Score between user specified in input and all other users who are matched by General Matching Feature. In order to receive this data, UserMatchingScoreService is used.

Second mapping is mapped to addUserQuestionExtendedAnswers() method. This method takes as input authenticated user and list of UserQuestionExtendedAnswersWeb objects, and does not return anything. This method is designed to get user’s answers for Extended Questions and push it UserQuestionExtendedAnswerService in order to store this data.

**3.2. Services.**

**3.2.1. AnswerSimilarityService**

This service is designed to process requests related to answers similarity values. It has one method getAnswerSimilarity(), which takes 2 AnswerExtended objects as input and return AnswerSimilarity object as output. This method is designed to validate AnswerExtended objects and get corresponding information from AnswerSimilarityRepository.

**3.2.2. CoreMatchingService**

This service is designed to implement User Matching Score calculation formula and contains 2 methods: evaluateUserMatchingScore() and evaluateAllUserMatchingScores(). Also this service contains value of Basic Matching Score, described in p.2.1.

First method is designed to calculate User Matching Score between 2 users using formula from p.2.1. It takes 2 User objects as input and returns UserMatchingScore object as output. Also it updates corresponding User Matching Score in the database using UserMatchingScoreRepository.

Second method is designed to calculate User Matching Scores between each user and all other users who are matched by General Matching Feature. The purpose of this method is to perform recalculation of all User Matching Scores in database and potentially could consume a lot of resources.

Also this service contains private methods:

calculateMatchingScore – loads all required information and calculates value of User Matching Score according to formula described in p.2.1

getAnswerSimilarityScore – calculates maximum answer similarity value between 2 lists of answers for Extended Questions given by user origin and matching user, described in p.2.4.

**3.2.3. QuestionExtendedService**

This service is designed to process requests related to Extended Questions. It has 2 methods: getQuestionsExtendedWithAnswers() and getQuestionImportanceList().

First method is designed to return the collection of all Extended Questions with corresponding answers. It takes Language object as input and returns the list of QuetionExtendedWithAnswersLocaleWeb objects as output. It retrieves all Extended Questions using QuestionExtendedRepository and corresponding answers using AnswerExtendedRepository.

Second method is designed to return list of all Question Importance values, it takes Language object as input and returns the list of QuestionImportanceDTO objects as output. This methods retrieves information using QuestionImportanceRepository.

**3.2.4. UserMatchingScoreService**

This service is designed to process all requests related to User Matching Score. It has 3 methods: getUserMatchingScore(), getUserMatchingScores() and deleteUserMatchingScores().

First method is designed to retrieve User Matching Score between 2 users. It takes 2 User objects as input and returns UserMatchingScore as output. This method tries to retrieve information from database using UserMatchingScoreRepository and in case if this value is null, it requests CoreMatchingService to calculate it via evaluateMatchingScore() method.

Second method is designed to get User Matching Scores between specific user and all other users who are matched by General Matching Feature. It takes User object as input and returns collection of pairs “userId”: “matchingScore” as output. This method calls getUserMatchingScore() for each found matching user.

Third method is designed to delete all User Matching Score information from database where specific user is present. It takes User object as input and returns number of deleted rows in database, using UserMatchingScoreRepository.

**3.2.5. UserQuestionExtendedAnswerService**

This service is designed to process all requests related to user’s answers for Extended Questions. It has 3 methods: getAllUserQuestionExtendedAnswers(), addUserQuestionExtendedAnswers() and getSelectedUserQuestionExtendedAnswers().

First method is designed to retrieve all user answers for Extended Questions. It takes User object as input and returns collection of pairs “QuestionExtended”: “list of UserQuestionExtendedAnswer objects”. This method retrieves information from database using UserQuestionExtendedAnswerRepository.

Second method is designed to store user answers for Extended Questions to database. It takes User object and list of UserQuestionExtendedAnswersWeb as input and does not return anything. Before adding this method deletes all information about previous user’s answers for Extended Questions and all User Matching Scores where specific user is present, using UserMatchingScoreService and UserQuestionExtendedAnswerRepository. Also this method calculates and stores maximum score described in p.2.1. This method saves answers for Extended Questions using UserQuestionExtendedAnswerRepository.

Third method is designed to retrieve all user answers for Extended Questions, but in comparison to getAllUserQuestionExtendedAnswers() it includes also Extended Questions, which user did not answer. It takes User object as input and returns list of UserQuestionExtendedAnswersWeb objects. Basically this method calls getAllUserQuestionExtendedAnswers() method, convert result into specified above object and adds not answered Extended Questions.

**3.3. Repositories**

**3.3.1. AnswerExtendedRepository**

This repository is designed process all requests related to answers for Extended Questions stored in database. It has 2 methods: getAllAnswersExtended() and getLocaleAnswersExtended().

First method does not take any input and returns list of all AnswerExtended objects storing in database in table “I\_ANSWER\_EXTENDED”.

Second method takes Language object as input and returns list of AnswerExtendedResponseDTO objects, which correspond to stored values in table “I\_ANSWER\_EXTENDED\_LOCALE” in specified language.

**3.3.2. QuestionExtendedRepository**

This repository is designed process all requests related to Extended Questions stored in database. It has 2 methods: getAllQuestionsExtended() and getLocaleQuestionsExtended().

First method does not take any input and returns list of all QuestionExtended objects storing in database in table “I\_QUESTION\_EXTENDED”.

Second method takes Language object as input and returns list of QuestionExtendedResponseDTO objects, which correspond to stored values in table “I\_QUESTION\_EXTENDED\_LOCALE” in specified language.

**3.3.3. QuestionImportanceRepository**

This repository is designed process all requests related to Question Importance stored in database. It has 2 methods: getAllQuestionImportance() and getLocaleQuestionImportance().

First method does not take any input and returns list of all QuestionImportance objects storing in database in table “I\_QUESTION\_IMPORTANCE”.

Second method takes Language object as input and returns list of QuestionImportanceDTO objects, which correspond to stored values in table “I\_QUESTION\_IMPORTANCE\_LOCALE” in specified language.

**3.3.4. AnswerSimilarityRepository**

This repository is designed process all requests related to the value of answers similarity stored in database. It has one method: getAnswerSimilarity(), which takes 2 AnswerExtended objects and returns AnswerSimilarity object which corresponds to a value of answer similarity between 2 specifies answers, stored in table “ANSWER\_SIMILARITY”.

**3.3.5. MaxUserMatchingScoreRepository**

This repository is designed process all requests related to maximum matching score stored in database. It has one method: getUserMaxMatchingScore(), which takes User object as input and returns MaxUserMatcingScore object which corresponds to a value of maximum matching score value described in p.2.1, stored in table “MAX\_USER\_MATCHING\_SCORE”.

**3.3.6. UserMatchingScoreRepository**

This repository is designed process all requests related to User Matching Score stored in database. It has 3 methods: getUserMatchingScore(), getAllUserMatchingScores() and deleteUserMatchingScores().

First method is designed to retrieve User Matching Score between 2 users from database. It takes 2 User objects as input and returns UserMatchingScore object as output stored in table “USER\_MATCHING\_SCORE”.

Second method is designed to get User Matching Scores from database where specific user is present. It takes User object as input and returns list of UserMatchingScore objects as output stored in table “USER\_MATCHING\_SCORE”, where user’s ID is present in field “USER\_ORIGIN\_ID”.

Third method is designed to delete all User Matching Score information from database where specific user is present. It takes User object as input and returns number of deleted rows from table “USER\_MATCHING\_SCORE”, where user’s ID is present in field “USER\_ORIGIN\_ID” or “USER\_TO\_MATCH\_ID”.

**3.3.7. UserQuestionExtendedAnswerRepository**

This repository is designed process all requests related to users answers for Extended Questions stored in database. It has 2 methods: getUserQuestionExtendedAnswer() and getAllUserQuestionExtendedAnswers().

First method takes User and QuestionExtended objects and returns list of UserQuestionExtendedAnswer objects, which correspond to chosen answers for specified Extended Question by specified user stored in table “USER\_EXTENDED\_QUESTION\_ANSWER”.

Second methods takes User object as input and returns list of UserQuestionExtendedAnswer objects, which correspond to chosen answers for all Extended Questions by specified user stored in table “USER\_EXTENDED\_QUESTION\_ANSWER”.

**3.4. Entities**

**3.4.1. AnswerExtended**

This entity is designed to describe answers for Extended Questions, stored in “I\_ANSWER\_EXTENDED” table and contains reference for corresponding QuestionExtended object.

**3.4.2. AnswerExtendedLocale**

This entity is designed to describe localized description of answers for Extended Questions, stored in “I\_ANSWER\_EXTENDED\_LOCALE” table and contains references for corresponding AnswerExtnded and Language objects.

**3.4.3. QuestionExtended**

This entity is designed to describe Extended Questions, stored in “I\_QUESTION\_EXTENDED” table.

**3.4.4. QuestionExtendedLocale**

This entity is designed to describe localized description of Extended Questions, stored in “I\_QUESTION\_EXTENDED\_LOCALE” table and contains references for corresponding QuestionExtended and Language objects.

**3.4.5. QuestionImportance**

This entity is designed to describe Question Importance, stored in “I\_QUESTION\_IMPORTANCE” table and contains numeric value called “weight”.

**3.4.6. QuestionImportanceLocale**

This entity is designed to describe localized description of Question Importance, stored in “I\_QUESTION\_IMPORTANCE\_LOCALE” table and contains references for corresponding QuestionImportance and Language objects.

**3.4.7. AnswerSimilarity**

This entity is designed to describe similarity between answers for Extended Questions, stored in “ANSWER\_SIMILARITY” table and contains references for 2 AnswerExtended objects and corresponding numeric value “similarityScore” which describes how similar are specified answers.

**3.4.8. MaxUserMatchingScore**

This entity is designed to describe maximum possible matching score for chosen by specific user answers for Extended Questions, described in p.2.1., stored in table “MAX\_USER\_MATCHING\_SCORE” and contains references for User object and corresponding numeric value “score”.

**3.4.9. UserMatchingScore**

This entity is designed to describe User Matching Score between 2 users stored in table “USER\_MATCHING\_SCORE” and contains references for 2 User objects and corresponding numeric value “score” calculated by CoreMatchingService according to formula described in p.2.1.

**3.4.10. UserQuestionExtendedAnswer**

This entity is designed to describe user’s answers for Extended Questions including selected Question Importance, stored in table “USER\_EXTENDED\_QUESTION\_ANSWER” and contains references for User, QuestionExtended, AnswerExtended and QuestionImportance objects.