Week 3 - Tasks 5&6 weekly report

Rémi Pichon - 19 february 2016

What has been implemented

Task 5

I implemented corresponding liking genres to compute a better similarity.

- Execute (updated)

add method `displayCoOccurringGenreFrequencyAndConfidence` : displays frequency for each co-occuring genres existing binome and the confidence *conf(X => Y)*
add method `displayGenrePercentageOfTransaction` : displays for each co-occuring genres found the result : *conf(X => Y) / (supp(!X and Y) / supp(!X))* which gives an hint on how much a genre Y is actually liked if X genre is liked and appears in the same movie.

- ExecuteTaskFive (created)

the simple runner which instantiate the CaseSimilarity using the improved overlapse taking into accounts the co-occuring genres.

- DatasetReader (updated)

add method `computeCoOccuringGenre` which find every existing co-occurring genres and successively perform [1] conf(X=>Y) = supp(X and Y) / supp (X) then [2] supp(!X and Y) / supp(!X) then [1] / [2] which will be used to compute the following similarity.

- OverlapCaseGenreCoOccurringSimilarity (created)

update genres feature similarity using liking similarity

- FeatureSimilarity (update)

add method `public static double liking(DatasetReader reader, final Set<String> genres1, final Set<String> genres2)` which use computed ‘how much is Y to be liked if X is liked” data from the reader and apply it againsts every target’s genres.

Task 6

I implemented a new case feature similarity based on the contents of all the reviews written by the users. This content recommendation is performed by removing stop words, stemming them and applying two different term frequency : Binary and Normalised Term Frequency - Inverse Document Frequency (TFIDF). This recommendation improves significantly the recommender system, by far the better of all previous improvements implemented. The only drawback is the computing time for case similarity which takes 15 minutes.

- Execute (udpated)

add method `displayAllMovieTfidf` : display the matrix TFIDF with one word per line and all concatenated review for a movie in column.

- ExecuteTaskSix : once Reader get every data, either compute then store in a file or restore from a file both Binary and TFIDF matrix. These two matrix take approx. 15 minutes to compute. To save time, `boolean restoreMatrixFromFile` can be set to true in order to retrieve the pre-computed data from the files `TFIDFMatrix.txt` and `binaryMatrix.txt` which are given in /data-set folder.

- MovieCase.java (updated)

add field `private String reviews;` which contains all concatenated reviews after stop word, stem words, lower case and without punctuation. Getter and setter are also provided

- DatasetReader (updated)

add `concatMovieReview` which for each movie concat all reviews, remove stop word, remove punctuation and stemm words.
add `computeBinaryIntoSparseMatrix` and `computeTfidfIntoSparseMatrix` which works almost the same way : for each word of each review of one movie, compute the content base similarity among all reviews of all movies.

- OverlapCaseCosineReviewSimilarity (created)

add feature similarity using case based's review with a cosine similarity function

- FeatureSimilarity (update)

add method `public static double cosine(DatasetReader reader, MovieCase candidate, MovieCase target)` in order to perform a cosine similarity between two movies. Read the matrix (binary or TFIDF depending on what have been computed in ExecuteTaskSix) and perform the cosine similarity for every common words on the two documents with all reviews for a movie.

- Stemmer and Stopwords (created)

these two classes has been forked from https://github.com/harryaskham/Twitter-L-LDA (which is open source and free to use) to remove stop words and stem them all.

Running instruction

There is one class ExecuteTaskFive.java with a main to run …..
There is one class ExecuteTaskSix.java with a main to run … with doesn't work because it takes too many times to compute.