Il recommender system realizzato utilizza l'algoritmo di Rocchio per creare un modello dell'utente.
Si utilizzano solo i documenti\film su cui l'utente ha dato un feedback positivo per creare il vettore prototipo andando a calcolare il centroide dei documenti su cui l'utente ha dato un voto positivo e calcolando poi la similarità tra quest'ultimo e i restanti film su cui l'utente non ha espresso alcun giudizio.

La classe "ConvertRatingFile" implementa il metodo main() per convertire i rating di ogni utente, descritti nel file ua.base, in formato binario (1 = like, 0 = dislike).

La classe "XmlIndexing" implemnta un metodo per parserizzare il file XML contenente le descrizioni dei vari film delle quali prendiamo in considerazione solo alcuni campi (titolo, generi, attori, registi, trama e id). Tale metodo restituisce una lista di oggetti di classe Movie, classe che modella le descrizioni dei film.

La classe "MovieIndexer" implementa il metodo main() per indicizzare con Lucene la lista dei film risultante dopo la parserizzazione del file XML. Coerentemente con i campi presi in considerazione durante la parserizzazione del file XML, ogni documento di Lucene inserito nell'indice si compone di diversi campi: id, titolo, genere, attori, registi, trama).

La classe "TfidfResolver" ha due metodi. Il primo di questi (getTfidf) si occupa di restituire una HashMap che ha come chiave l'id di un film e come valore il term vector (che a sua volta è stato implementato tramite una HashMap con chiave pari ad un termine e con valore il suo tfidf). Il secondo metodo (normalize) invece implementa l'algoritmo per normalizzare un term vector.

La classe "TfidfLikes" implementa metodi per ritrovare i term vector dei film su cui l'utente ha dato un feedback positivo. In particolare il metodo getIdLikes restituisce la lista degli id relativi ai film su cui l'utente ha dato un voto positivo.
Il metodo getVectLikes restituisce una HashMap che ha come chiave l'id di un film che piace all'utente e come chiave il term vector del relativo film, implementato ancora una volta con una HashMap.
Il metodo getRocchioLikes restituisce il vettore prototipo dei film che piacciono all'utente, implementato con una HashMap come in precedenza.

E' stata implementata anche la classe "CosineSimilarity" che ha un metodo per il calcolo della similarità tra due vettori, ancora una volta implementati con HashMap.

La classe "RocchioSimilarity" modella invece la similarità di un film con il vettore prototipo calcolato con l'algoritmo di Rocchio. I due campi che la compongono sono l'id del documento e un double che rappresenta il valore di similarità con il prototipo.

Infine è stata implementata la classe "MainClass" che implementa il metodo main opportunamente parametrizzato. Il metodo main andrà a calcolare il vettore prototipo dei like dell'utente, calcolerà la similarità fra tale vettore e tutti i film non ancora votati dall'utente, ordinerà tali film sulla base della similarità in maniera decrescente e stamperà a video i primi k risultati.

Si è cercato di commentare al meglio il codice, inserendo prima di ogni main delle righe che spiegano quali sono gli argomenti da inserire nella run configuration.

Prima di avviare il main della classe "MainClass" di raccomandazione bisognerà avviare rispettivamente prima il main della classe "ConvertRatingFile" e poi quello della classe "MovieIndexer".