Framework for recommendation system

Abstract-

Recommendation systems attempt to predict the preference or rating that a user would give to an item.

Knowledge discovery techniques can be applied to the problem of making personalized recommendations about items or information during a user's visit to a website. Collaborative Filtering algorithms give recommendations to a user based on the ratings of other users in the system.

Traditional collaborative filtering algorithms face issues such as scalability, sparsity and cold start.

We are here to bring a common platform for including libraries and multiple toolsets to combine and ease developer's work.

A number of frameworks for Recommender Systems (RS) have been proposed by the scientific community, involving different programming languages, such as Java, C\#.

Python, among others. However, most of them lack an integrated environment containing clustering and ensemble approaches which are capable to improve recommendation accuracy.

On the one hand, clustering may support developers to pre-process their data to optimize or extend recommender algorithms.

On the other hand, ensemble approaches are efficient tools to combine different types of data in a personalized way.

Approach	Item Recommender	Rating Prediction
Neighborhood-Based	User KNN, User Attr KNN, Item KNN, Item Attr KNN, Content Based	User KNN, User Attr KNN, Item KNN, Item Attr KNN
Matrix Factorization	BPR MF	MF, Item-MFMS, SVD, SVD++, GSVD++, Item NSVD1, User NSVD1
Non-Personalized	Most Popular, Random	Most Popular, Random
Ensemble	BPR Learning, Tag Based, Average Based	-
Clustering-based	PaCo, Group-based	-

The framework is now implemented in Python 3 and it addresses two common scenarios in recommender systems: rating prediction and item recommendation, using explicit, implicit or both types of feedback in several recommender strategies.

```
easy_install CaseRecommender
```

Usage-

For divide our dataset using Fold Cross Validation:

```
from caserec.utils.split_database import SplitDatabase

SplitDatabase(input_file=dataset, dir_folds=dir_path,
n_splits=10).k_fold_cross_validation()
```

Run Item Recommendation Algorithm (E.g: ItemKNN)

```
from caserec.recommenders.item_recommendation.itemknn import ItemKNN

ItemKNN(train_file, test_file).compute()
```

Run Rating Prediction Algorithm (E.g: ItemKNN)

```
from caserec.recommenders.rating_prediction.itemknn import ItemKNN

ItemKNN(train_file, test_file).compute()
```

Evaluate Ranking (Prec@N, Recall@N, NDCG@, Map@N and Map Total)

```
from caserec.evaluation.item_recommendation import
ItemRecommendationEvaluation
 ItemRecommendationEvaluation().evaluate_with_files(predictions_file,
test_file)
```

Evaluate Ranking (MAE and RMSE)

```
from caserec.evaluation.rating_prediction import
RatingPredictionEvaluation

RatingPredictionEvaluation().evaluate_with_files(predictions_file,
test_file)
```

Run ItemKNN in Fold Cross Validation Approach

```
from caserec.recommenders.item_recommendation.itemknn import ItemKNN
from caserec.utils.cross_validation import CrossValidation

## Cross Validation
recommender = ItemKNN(as_binary=True)
CrossValidation(input_file=db, recommender=recommender,
dir_folds=folds_path, header=1, k_folds=5).compute()
```

During my studies, we also built a repository of a topic-centric public data sources in high quality for RS. They are collected and tidied from Stack Overflow, articles, recommender sites and academic experiments. Most of the datasets presented in the repository are free, having open source licenses.

Input Data-

The framework allows developers to deal with different datasets and not having to develop their own programs to execute recommender functions. The input of algorithms expects the data to be in a simple text format:

user_id item_id feedback

where user_id and item_id are integers referring to users and items IDs, respectively, and feedback is a number expressing how much the user likes an item or binary interaction. The separator between the values can be either spaces, tabs, or commas. If there are more than three columns, all additional columns are ignored. For example, here is a sample of data from the ML100k dataset:

```
196 242 3
186 302 3
22  377 1
244 51  2
166 346 1
298 474 4
115 265 2
253 465 5
305 451 3
6   86  3
62  257 2
```

Case Study

	Shrek	Snow-white	Superman
Alice	Like	Like	Dislike
Bob	?	Dislike	Like
Chris	Like	Like	Dislike
Jon	Like	Like	?

1) As we can see from the table when the data is increased the model will be trained more .

2) For example here Bob can like or dislike the movie shriek based on how much data we provide.If we are taking the data only for shriek movie we can see everyone liked it so Bob will also like it .

3) But of we take a larger dataset we can see that the choice of Bob is exactly opposite of others.So he will dislike the movie shriek

4) So in smaller and bigger model the result for Bob changed.

1)That when one builds a recommendation system, they go for 2-3 approaches and result may vary,

2) but if you have a framework that includes most of the algorithm and tools already,

3) he or she can see immediately with multiple algorithms on how results are coming saves time and coding

a

Conclusion-

The goal of Case Recommender is to integrate and facilitate the experiments and development of new recommender techniques for different domains. Our framework contains a recommender engine, that contains content-based, collaborative and hybrid filtering approaches for rating prediction and item recommendation scenarios. In addition, the framework contains ensemble and clustering algorithms, validation and evaluation metrics to improve and measure the quality of the recommendation.

Thank You

a