Case Recommender: A Recommender Framework

Arthur F. da Costa and Marcelo G. Manzato
Institute of Mathematics and Computer Science
University of São Paulo
São Carlos, SP, Brazil
{fortes,mmanzato}@icmc.usp.br

ABSTRACT

Case Recommender is a Python implementation of a number of popular content-based and collaborative recommendation algorithms which use implicit and explicit feedback. The framework aims to provide a rich set of components from which developers can construct a customized recommender system. One differential of the framework is the possibility to combine multiple recommenders in a post-processing ensemble approach to produce more accurate results. Case Recommender can be used for rating prediction and item recommendation tasks, and it includes different metrics and procedures for validation and evaluation.

Keywords

Recommender Systems; Framework

1. INTRODUCTION

The growth of on-line content available to users has made finding and consuming relevant items a challenge that users have to deal with everyday. In response to this problem, recommender systems have been created, which are an information filtering technology that can be used to predict preference ratings of items, not currently rated by the user, and/or to output a personalized ranking of items/recommendations that are likely to be of interest to the user [7].

In order to obtain such interests, profiling mechanisms have been developed, which consist of acquiring, representing and maintaining pieces of information relevant (and/or irrelevant) to the user. In the particular case of obtaining user's preferences, the three most known techniques are based on explicit feedback, implicit feedback and hybrid approaches. Implicit information is collected indirectly during user navigation with the system while visiting a page, mouse movement and clicks on various links of interest [4]. Regarding explicit feedback, the data is intentionally provided, i.e., the users express themselves in some direct way (e.g. filling in forms or rating a content). This type of information is considered more reliable, since the user is the one who provides the topics of interest, but the cost of this procedure is the effort of the individual, who is not always willing to cooperate with the system [7]. Finally, the hybrid approach consists of applying the implicit and explicit feedback together, in order to obtain a greater number of user information [7].

Traditionally, recommender systems employ filtering techniques and machine learning information to generate appropriate recommendations to the user's interests from the representation of his profile. However, other techniques, such as Neural Networks, Bayesian Networks and Association Rules, are also used in the filtering process [2]. The most used types of filtering are currently: Content-Based (CBF), responsible for selecting information based on content filtering of elements, e.g., e-mail messages filtered out as trash for containing unwanted words; Collaborative Filtering (CF), based on the relationship between people and their subjective regarding the information to be filtered. The selection of electronic messages based on the relationship between sender and recipient is an example of that. Besides, there is a Hybrid approach that combines the content-based and the collaborative filtering methods. [7].

Although there are useful libraries which can be used to implement, evaluate and extend classical recommenders, such as Apache Mahout[1], MyMediaLite[2], LensKit[3], among others, the viability of a framework which is able to contain multiple recommender algorithms available using a variety of data sources is unknown. Such framework, indeed, should be able to deal with the effects of using large volumes of data considering hardware configuration, and the impact of model updates in the recommender performance.

In this paper we describe a new framework called Case Recommender, which contains a variety of content-based and collaborative recommender algorithms, as well as ensembling approaches for combining multiple algorithms and data sources. In addition, it provides a set of popular evaluation methods and metrics for rating prediction and item recommendation. As a result, it can be used to conduct fair and comprehensive comparisons between different recommendation algorithms. The framework is published in PyPi[4], a international repository of software for the Python programming language, as a GNU GPLv3 License, making it easy for third parties to contribute with additional implementations and features.

This paper is structured as follows: Section 2 addresses the related work; Section 3 describes the framework pro-

In: Workshop de Ferramentas e Aplicações (WFA), 15., 2016, Teresina. Anais do XXII Simpósio Brasileiro de Sistemas Multimídia e Web. Porto Alegre: Sociedade Brasileira de Computação, 2016. v. 2.

ISBN: 978-85-7669-332-1

[1] https://mahout.apache.org/
[2] http://www.mymedialite.net/
[3] http://lenskit.org/
[4] https://pypi.python.org/pypi/CaseRecommender/

posed in this work; Section 4 reports Case Recommender architecture; Section 5 presents details of the license and distribution of the code; finally, Section 6 is devoted to final remarks and future works.

2. RELATED WORK

A number of frameworks for recommender systems have been proposed by the scientific community, involving different programming languages, such as C/C++, Java, C#, Python, among others. One of the most popular and complete frameworks is the open-source recommendation library MyMediaLite, which has been developed since 2011 and contains a variety of recommender algorithms for rating prediction and item recommendation. However, it lacks content-based filtering approaches, besides its design which embarrasses new developers to extend it with new features.

Other libraries, such as EasyRec[5], Mahout, LensKit and RiVal[6] have also been proposed, but most of them contain only a small set of recommendation approaches. Similarly to MyMediaLite, most of them do not provide content-based recommenders. In addition, none of them provides ensemble techniques that allow developers to combine multiple recommenders. Table 2 summarizes the features available in each considered recommender library.

In this sense, Case Recommender was developed to provide flexibility and extensibility in research environments, while maintaining high performance. Its primary concern is to maximize usefulness for research and education, instead of large-scale commercial operations. The framework is also designed to support a wide variety of recommendation approaches, including content-based and collaborative filtering.

3. CASE RECOMMENDER

Case Recommender addresses two common scenarios in content-based and collaborative filtering: rating prediction and item recommendation, using either explicit and/or implicit feedback. It offers state-of-the-art algorithms for these two tasks and validation and evaluation metrics to measure the recommender algorithms accuracy. Our framework is implemented in Python with scientific environments for numerical applications such as Scipy[7] and NumPy[8]. Using these free and open-source libraries, it can be used on the most popular operating systems. To install Case Recommender on Mac OS X or Linux, chances are that one of the following two commands will work for developers and users (with admin privileges):

easy_install CaseRecommender

or alternatively:

pip install CaseRecommender

Windows users who do not have the easy_install command must first install it before installing the library. For this, they need to check the pip and setuptools on Windows tutorial[9] for more information about how to do that. Once they have it installed, it is possible to run the same commands above.

An important feature in the design of our framework was to enable the computation of recommendations in large-scale. Case Recommender is currently being rewritten to support optimized calculations using known Python scientific libraries. Another feature is to support sparse and large datasets in a way that there is as little as possible overhead for storing data and intermediate results. Moreover, our framework aims to support scaling in recommender systems in order to build high-scale, dynamic and fast recommendations over simple calls.

Another feature of the framework is its extensibility and flexibility, as developers can implement new recommendation algorithms while using the available data structures and routines. According to the application scenario, developers can choose between using one of the available recommender algorithms, or combining multiple recommendations using one of the available ensemble techniques [3, 2]. Table 1 shows the recommender algorithms in the framework.

Table 1: Case Recommender Algorithms.

Approach	Item Recommender	Rating Prediction
Neighborhood Based	UserKNN	UserKNN
	User Attr KNN	User Attr KNN
	ItemKNN	ItemKNN
	Item Attr KNN	Item Attr KNN
Matrix Factorization	BPR MF	MF
		SVD
		ItemNSVD1
		UserNSVD1
Ensemble	Tag-Based	-
	Average-Based	
	BPR Learning	

Case Recommender contains a generic interface for recommender systems implementations, among them the collaborative and content-based filtering approaches such as neighborhood-based (NB) and matrix factorization (MF) models, which are already available for use. The recommender interfaces can be easily combined with more than 15 different pairwise metrics already implemented, like cosine, tanimoto, pearson and euclidean, using Scipy basic optimized functions[10]. Moreover, it offers support for using similarity functions such as user-to-user or item-to-item and allows easy integration with different input domains like databases, text files or python libraries.

4. ARCHITECTURE

The proposed framework consists of a set of classes and methods for generating and evaluating recommendations in rating prediction and item recommendation scenarios. Figure 1 illustrates all components involved.

There are three main tasks in Case Recommender: data representation, recommendation and evaluation. The following subsections detail each of them.

[5]http://www.easyrec.org/
[6]http://rival.recommenders.net/
[7]https://www.scipy.org/
[8]http://www.numpy.org/
[9]http://docs.python-guide.org/en/latest/starting/install/win/
[10]http://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.spatial.distance.pdist.html

Table 2: Comparison of free/open source recommender system toolkits.

Toolkits	Case Recommender	EasyRec	Mathout	LensKit	RiVal	MyMediaLite	Crab
Version	0.0.6	0.99	0.12.2	2.2.1	0.2	3.11	0.1
Last update	Jul 2016	Nov 2013	Jun 2016	Nov 2015	Aug 2015	Dec 2015	Jan 2012
Actively Developed	✓	-	✓	✓	-	✓	-
License	GPL 3	GPL 3	Apache 2.0	LGPL 2	Apache 2.0	GPL 3	BSD
Language	Python	Java	Java	Java	Java	C#	Python
Scalable	✓	-	✓	✓	-	✓	✓
Rating Prediction	✓	✓	✓	✓	✓	✓	✓
Item Recommender	✓	-	-	✓	-	✓	✓
CF Techniques	✓	✓	✓	✓	✓	✓	✓
CBF Techniques	✓	-	-	-	-	-	-
Ensemble Techniques	✓	-	-	-	-	-	-

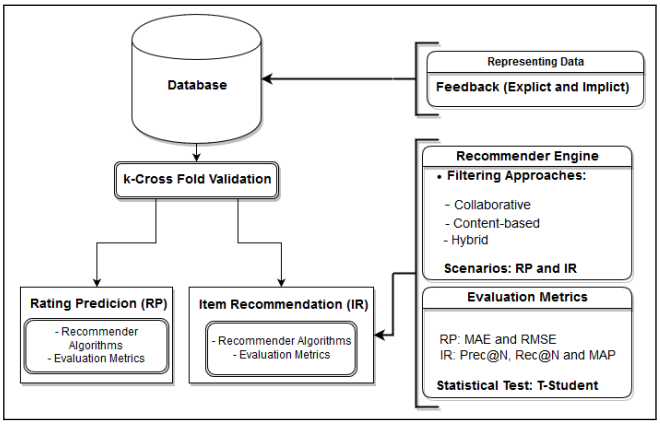

Figure 1: Case Recommender Architecture.

4.1 Representing Data

Case Recommender is mainly a framework, designed to be used by other applications. We provide command-line interfaces, classes and functions that offer most of framework's functionality. The framework allows developers to deal with different datasets and not having to develop their own programs to execute recommender functions.

The input of rating prediction algorithms expects the data to be in a simple text format:

u_id i_id rating

where *u_id* and *i_id* are integers referring to users and items IDs, respectively, and *rating* is a floating-point number expressing how much the user likes an item. The separator between the values can be either spaces, tabs, or commas. If there are more than three columns, all additional columns are ignored.

The item recommendation approaches behave similarly to the rating prediction approaches. The main difference is that in most of the item recommendation algorithms the user can omit or ignore the third column, which corresponds to the feedback value. However, there are some algorithms that may or not use the feedback value in both scenarios.

4.2 Recommender Engines

The proposed framework contains a recommender engine composed of several algorithms described in the literature, such as user and item-based kNN and matrix factorization. It provides an assortment of components that may be plugged together and customized to create an ideal recommender for a particular domain.

4.2.1 Rating Prediction

Ratings are a popular kind of explicit feedback. Users assess how much they like a given item (e.g. a movie or a news article) on a predefined scale, e.g. 1 to 5, where 5 could mean that the user likes the item very much, whereas 1 means the user strongly dislikes the item [4]. Rating prediction algorithms estimate unknown ratings from a given set of known ratings and possibly additional data like user or item attributes. The predicted ratings can then indicate to users how much they will like an item, or the system can suggest items with high predicted ratings. Our available prediction algorithms are mentioned in Table 1.

4.2.2 Item Recommendation

In the item recommendation task, the system constructs a list of items which are more likely to be preferred by a given user. Usually in the item recommendation scenario the user's feedback is implicit, which means that his preferences on items are unobservable [5]. For example, in the movie recommendation task, the number of times a user watches a movie is observable (i.e., implicit feedback) but not his explicit preference rating about the movie (i.e., explicit feedback). Recommendation models are then built based on some pre-defined preference assumptions, e.g., the more times a user watches a movie, the more he/ she likes it. Such assumptions may not accurately describe users' preferences. Moreover, the observed user-item interaction data are generally sparse, which makes the preference modeling even more challenging. As a result, existing solutions often deliver unsatisfactory recommendation accuracies.

4.2.3 Ensemble Approaches

Case Recommender provides an extensible platform for ensemble approaches which allow developers to combine different recommender algorithms using different input data. Its flexibility allows developers to build complex architectures for better accuracy depending on the application scenario. Such ensemble approaches, as they are integrated in the framework, allow their evaulation with fair comparisons against individual filtering approaches. Our implementation provide a clear interface, where the ensemble strategies receive the results of previous probe runs (whether required),

and the set of results from previous trained recommenders.

4.3 Validation and Evaluation Metrics

Case Recommender contains routines for recommendation evaluation, including measures such as root mean square error (RMSE) and mean average error (MAE) for the rating prediction task. For the item recommendation task, the available metrics are precision-at-N (prec@N), recall-at-N (recall@N) and mean average precision (MAP). For both scenarios, internal K-fold cross-validation is supported to validate the experiments.

The framework also contains the All But One [1] protocol for the construction of the ground truth using K-fold-cross-validation. Given the data set, the system randomly divides it into the same K subsets and for each sample it uses $n-1$, these subsets of data for training and the rest for testing. The training set t_r is used to test the recommendation technique and in the test set T_e the system randomly separates an item for each user to create the truth set H. After that, the remaining items form the set of observable O, which is used to test the algorithm.

In order to compare the results with statistical significance, the two-sided paired t-test [6] is available in Case Recommender.

4.4 Usage

For each of the two recommendation scenarios, Case Recommender comes with a command-line instructions that allow users to train and evaluate all available recommenders on data provided in text files. When a new recommender is added into the framework, it is automatically detected, so developers do not have to manually add new features into the programs, only use the command *import* in Python. The main instructions are available at Github[11].

5. LICENSE AND DISTRIBUTION

Case Recommender is freely available under GNU GPLv3 license. The GPL grants the recipients of a computer program the rights of the Free Software Definition[12] and uses copyleft to ensure the freedoms are preserved whenever the work is distributed, even when the work is changed or added to. The GPL is a copyleft license, which means that derived works can only be distributed under the same license terms.

We chose this license because it allows programs to be distributed and reused, keeping, however, the author's rights in order to not allow this information to be used in a way that limits the original freedoms.

Our project is hosted at Github repository and it is available for the scientific community to use, test and contribute. Future releases are planned which will include more features and an evaluation tool with several plots and graphs to help developers to better understand the behavior of their recommender algorithms. The source code is freely available at Github[11].

6. FINAL REMARKS

In this paper, we presented a framework to recommender systems, called Case Recommender. The goal of this framework is to integrate and facilitate the experiments and development of new recommender techniques for different domains. Case Recommender contains a recommender engine, that contains content-based and collaborative approaches based on neighborhood and matrix factorization models for rating prediction and item recommendation scenarios. In addition, the framework contains ensemble algorithms, validation and evaluation metrics to improve and measure the quality of the recommendation.

The main advantages of our framework are extensibility and flexibility, once it enables developers to use and develop different recommender algorithms in both scenarios of recommendation and also allows different ways of usage of methods and classes during the recommendation process. Moreover, the framework stands out for containing recent content-based filtering algorithms and ensemble approaches, differently from the other recommender toolkits cited in this paper.

We will continue Case Recommender's development in several directions: implementing new recommendation algorithms, evaluation and validation metrics and porting the framework for other programming languages. We also plan to add additional recommendation tasks and types of input, e.g. item prediction from other kinds of implicit feedback like viewing times or click counts, or tag recommendation.

[11]https://github.com/ArthurFortes/CaseRecommender
[12]https://www.gnu.org/licenses/

7. ACKNOWLEDGMENTS

We would like to acknowledge CAPES and CNPq for the financial support.

8. REFERENCES

[1] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, pages 43–52, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[2] A. F. da Costa and M. G. Manzato. Exploiting multimodal interactions in recommender systems with ensemble algorithms. *Information Systems*, 56:120 – 132, 2016.

[3] A. da Costa Fortes and M. G. Manzato. Ensemble learning in recommender systems: Combining multiple user interactions for ranking personalization. In *Proceedings of the 20th Brazilian Symposium on Multimedia and the Web*, WebMedia '14, pages 47–54, New York, NY, USA, 2014. ACM.

[4] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. MyMediaLite: A free recommender system library. In *Proceedings of the 5th ACM Conference on Recommender Systems*, 2011.

[5] N. N. Liu, M. Zhao, and Q. Yang. Probabilistic latent preference analysis for collaborative filtering. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 759–766, New York, NY, USA, 2009. ACM.

[6] T. M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[7] F. R., L. R., and B. S. Introduction to recommender systems handbook. In *Recommender Systems Handbook*, pages 1–35. Springer US, 2011.