CARSKit: A Java-Based Context-aware Recommendation Engine

Yong Zheng
Center for Web Intelligence
DePaul University
Chicago, Illinois, 60604, USA
yzheng8@cs.depaul.edu

Bamshad Mobasher
Center for Web Intelligence
DePaul University
Chicago, Illinois, 60604, USA
mobasher@cs.depaul.edu

Robin Burke
Center for Web Intelligence
DePaul University
Chicago, Illinois, 60604, USA
rburke@cs.depaul.edu

Abstract—Recommender system has been demonstrated as one of the most useful tools to assist users’ decision makings. Several recommendation algorithms have been developed and implemented by both commercial and open-source recommendation libraries. Context-aware recommender system (CARS) emerged as a novel research direction during the past decade and many contextual recommendation algorithms have been proposed. Unfortunately, no recommendation engines start to embed those algorithms in their kits, due to the special characteristics of the data format and processing methods in the domain of CARS. This paper introduces an open-source Java-based context-aware recommendation engine named as CARSKit which is recognized as the 1st open source recommendation library specifically designed for CARS. It implements the state-of-the-art context-aware recommendation algorithms, and we will showcase the ease with which CARSKit allows recommenders to be configured and evaluated in this demo.

I. INTRODUCTION AND MOTIVATIONS

Recommender systems (RS) have been used to successfully address the information overload problem by providing recommendations to the end users. In recent years, researchers have realized the importance of contexts and try to incorporate contexts into RS to build context-aware recommender systems (CARS) which are able to adapt recommendations to users’ contextual situations.

Context is defined as any information that can be used to characterize any entity [4]. In CARS, contexts are usually selected from the dynamic attributes of the activity itself [9]. For example, user may choose a different type of movie if he or she is going to watch the movie with kids. For example, user may choose a different type of movie if he or selected from the dynamic attributes of the activity itself [9]. Unfortunately, no recommendation engines start to embed those algorithms in their kits, due to the special characteristics of the data format and processing methods in this domain. Researchers have to compile those algorithms by themselves again and again, which further impedes the development progress of CARS.

Therefore, it is badly in need of such a recommendation library to not only meet the basic design and evaluation purpose, but also boost the further development of context-aware recommendations. We introduce CARSKit—an open-source Java-based context-aware recommendation engine which implements the state-of-the-art contextual recommendation algorithms and provides a standard platform for evaluations and practical use. As far as we know, it is the 1st open source library specifically designed for CARS and it is able to provide flexible configurations and compatible spaces for expansion with new algorithms.

II. RELATED WORK

There are at least three reasons to develop a recommendation engine: 1). classical recommendation algorithms are the frequent ones to be evaluated in practice. A recommendation library is able to provide standard implementations and avoid repeatedly compiling those algorithms from time to time; 2). it is also a standard platform for benchmark or evaluations; 3). it is also helpful for both research purpose and industrial practice, as well as educational tools in teaching and learning.

During the past decades, tons of recommendation algorithms have been proposed and several recommendation engines, such as Mahout1, Duine2, Cofi3, EasyRec4, GraphLab Create5, LensKit6, LibRec7, MyMediaLite8, started to embed those algorithms into their kits. Here, we select the top popular ones and provide a brief comparison shown in Table I.

Mahout and GraphLab Create are two large-scale data mining and machine learning platforms, where GraphLab was an open-source library, but currently it was developed as one of DATO’s commercial products. MyMediaLite was one of the most popular recommendation libraries but it was no

1Mahout, http://mahout.apache.org/
2Duine, http://www.duineframework.org/
3Cofi, http://www.nongnu.org/cofi/
4EasyRec, http://easyrec.org/
5GraphLab Create, https://dato.com/products/create/
6LensKit, http://lenskit.org/
7LibRec, http://www.librec.net/
8MyMediaLite, http://www.mymedielite.net/
9DATO, https://dato.com/
TABLE I: Comparison Among Selected Recommendation Engines

<table>
<thead>
<tr>
<th>Initial release (date)</th>
<th>Mahout</th>
<th>GraphLab Create</th>
<th>MyMediaLite</th>
<th>EasyRec</th>
<th>LensKit</th>
<th>LibRec</th>
<th>CARSKit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latest version</td>
<td>05/2015</td>
<td>2015</td>
<td>09/2013</td>
<td>05/2013</td>
<td>05/2015</td>
<td>06/2015</td>
<td>09/2015</td>
</tr>
</tbody>
</table>

| Active updates          | Yes    | Yes             | No          | No      | Yes    | Yes   | Yes    |
| License                 | Apache | BSD, AGPL       | GPL         | N/A     | GPL    | GPL   | GPL    |
| Programming language    | Java   | Python, C++     | C#          | WebService | Java  | Java  | Java   |
| Running platform        | JVM    | Any             | NET         | WebApp  | JVM    | JVM   | JVM    |

| Command line            | Yes    | Yes             | Yes         | No      | Yes    | Yes   | Yes    |
| Documentation           | Yes    | Yes             | Yes         | Yes     | Yes    | Yes   | Yes    |
| Online updates          | Yes    | Yes             | Yes         | No      | No     | No    | No     |
| Algorithm diversity     | Classical| Classical | State-of-the-art | Classical | Classical | State-of-the-art | CARS |
| Distributed computing   | Partial | Yes            | No          | No      | No     | No    | No     |
| Open source             | Yes    | Partial         | Yes         | Yes     | Yes    | Yes   | Yes    |

| License                  | Apache | BSD, AGPL       | GPL         | N/A     | GPL    | GPL   | GPL    |
| Programming language    | Java   | Python, C++     | C#          | WebService | Java  | Java  | Java   |
| Running platform        | JVM    | Any             | NET         | WebApp  | JVM    | JVM   | JVM    |

TABLE II: Loose Format

<table>
<thead>
<tr>
<th>UserID</th>
<th>ItemID</th>
<th>Rating</th>
<th>Context</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>T1</td>
<td>3</td>
<td>Location</td>
<td>Weekend</td>
</tr>
<tr>
<td>U2</td>
<td>T2</td>
<td>4</td>
<td>Time</td>
<td>Weekday</td>
</tr>
</tbody>
</table>

TABLE III: Compact Format

<table>
<thead>
<tr>
<th>UserID</th>
<th>ItemID</th>
<th>Rating</th>
<th>Time</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>T1</td>
<td>4</td>
<td>Weekend</td>
<td>Work</td>
</tr>
<tr>
<td>U2</td>
<td>T2</td>
<td>2</td>
<td>Weekday</td>
<td>Work</td>
</tr>
</tbody>
</table>

longer updated for a while, where Rival\textsuperscript{10} and WrapRec\textsuperscript{11} were recently designed as a wrapper of MyMediaLite for standard benchmark and evaluations. They aim to provide a simple and unified way to create evaluation results comparable across different recommendation frameworks. EasyRec is a special one designed as a Web service deliver to the Web applications, but it was outdated and not updated anymore. LensKit and LibRec are two modern java-based recommendation libraries, where LensKit only provides the implementation of classical recommendation algorithms, such as user-based and item-based kNN collaborative filtering, slope one recommender, and matrix factorization. In fact, only MyMediaLite and LibRec are able to provide the state-of-the-art recommendation algorithms in addition to those classical ones, and LibRec runs faster based on a benchmark evaluation \cite{5}.

In contrast to those existing recommendation engines, CARSKit is specifically designed for context-aware recommendations, where the state-of-the-art recommendation algorithms in CARS are implemented in a way of easy-configuration and flexible-expansion.

III. CARSKit: Design and Features

CARSKit\textsuperscript{12} is an open-source free software, where it can be used, modified and distributed under the terms of the GNU General Public License. In this section, we will introduce the specific design and features in CARSKit.

A. Architecture and Design

CARSKit provides a flexible architecture so that it is easy to expand the scope of context-aware recommendation algorithms and provides spaces to develop new algorithms in the future. The whole design can be depicted by the Figure 1.

The workflow is straightforward in our design: different recommendation algorithms are the specific implementations and extensions from the generic interfaces where the shared and common functions are defined, such as rating or score prediction for a user on one item in a specific context. Evaluations for rating predictions and top-$N$ recommendations are embedded into the Recommender.

\textsuperscript{10}Rival, http://rival.recommenders.net/

\textsuperscript{11}WrapRec, https://github.com/babakx/WrapRec/

\textsuperscript{12}CARSKit, https://github.com/irecsys/CARSKit/

B. Features

Due to the special characteristics of the data format and processing methods in CARS, there are several challenges in the design. In this section, we will introduce the key features specifically designed for CARS, which can further be stated from those aspects: data transformer, data structure, recommendation algorithms, configuration and evaluations.

1) Data Transformer: In traditional RS, the rating prediction problem begins with a two dimensional matrix of ratings: $Users \times Items \rightarrow Ratings$. In CARS, contexts are considered as additional inputs, which results in a multidimensional rating function: $Users \times Items \times Contexts \rightarrow Ratings$ \cite{1}.

This special data format brings challenges to build such a recommendation kit, where the original structure in other recommendation engines may not be easily reused for context-aware recommendation.

Before the discussion of data structure, we introduce the data transformer here. Usually, the contextual rating data can be stored in two formats: loose format and compact format, as shown in tables below.

TABLE II: Loose Format

<table>
<thead>
<tr>
<th>UserID</th>
<th>ItemID</th>
<th>Rating</th>
<th>Context</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>T1</td>
<td>3</td>
<td>Location</td>
<td>Weekend</td>
</tr>
<tr>
<td>U2</td>
<td>T2</td>
<td>4</td>
<td>Time</td>
<td>Weekday</td>
</tr>
</tbody>
</table>

TABLE III: Compact Format

<table>
<thead>
<tr>
<th>UserID</th>
<th>ItemID</th>
<th>Rating</th>
<th>Time</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>T1</td>
<td>4</td>
<td>Weekend</td>
<td>Work</td>
</tr>
<tr>
<td>U2</td>
<td>T2</td>
<td>2</td>
<td>Weekday</td>
<td>Work</td>
</tr>
</tbody>
</table>
contextual rating profiles in the loose format but four rating profiles in the compact format in this example.

<table>
<thead>
<tr>
<th>UserID</th>
<th>ItemID</th>
<th>Rating</th>
<th>Time:Weekend</th>
<th>Time:Weekday</th>
<th>Location:Home</th>
<th>Location:Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>T1</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>U2</td>
<td>T2</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>U1</td>
<td>T1</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**TABLE IV: Binary Format**

Most contextual information is in shape of categorical data. Both the loose and compact format will increase storage pressure and computational costs. In CARSKit, we store contextual rating in a binary format as shown in Table IV, which is able to significantly boost the running performance. To assist the end users to prepare the rating data, we provide two methods TransformationFromLooseToBinary and TransformationFromCompactToBinary as the data transformer in our toolkit.

2) Data Structure: Since contexts are considered as additional inputs, the data structure becomes the most important part which directly leaves an impact on the flexibility and running performance.

There are two factors involved in designing the data structure: data storage and data operations. Intuitively, the context-aware data can be represented in a N-dimensional space as tensors, where each context dimension is considered as an individual dimension in the rating space. In this case, we build SparseTensor and TensorEntry to record the indices of each user, item, context dimension, and the associated rating. This structure is good for the context-aware recommendation algorithms using N-dimensional operations, such as the multiverse recommendation algorithm described in [6].

Meantime, there are many more contextual recommendation algorithms exploring the dependencies between contexts and user/item dimensions, where two-dimensional operation is still the most frequent one adopted in those algorithms. In this case, we continue to use SparseMatrix and SparseVector which are the well-recognized and good-efficient data representations in existing recommendation libraries, such as LensKit and LibRec. SparseMatrix uses the compressed row and column storage\(^\text{13}\) which was also demonstrated to boost the running efficiency in the design of LibRec [5].

3) Recommendation Algorithms: As mentioned before, CARSKit is the first library specifically designed for CARS. As shown in Figure 1, we divide the contextual algorithms into two categories: transformation algorithms and adaptation algorithms.

The transformation algorithms try to pre-process the data and convert the contextual data set to a 2-dimensional rating matrix which only contains users, items and ratings, so that any traditional recommendation algorithms can be applied to. One of the most effective techniques falling into this category is the context-aware splitting approaches [3], [10].

The adaptation algorithms directly incorporate contexts into the prediction function. There are two subcategories involved: independent modeling (e.g., TF [6]) which assumes contexts are independent with users (and items), and dependent modeling which exploits the dependencies among users, items and contexts, such as CAMF [2] and contextual sparse linear method (CSLIM) [12], [13]. Dependent modeling can be built in two ways: by modeling contextual rating deviations [2], [13] and by learning context similarities [14], [15]. Factorization machines (FM) [7] is a finer-grained algorithm which exploits pairwise relationships in its learning process. Among those algorithms, TF and CAMF are two popular ones which have been recognized as the standard baselines in CARS.

In addition to those state-of-the-art contextual recommendation algorithms, we also include some traditional recommendation algorithms in the package baseline. We did not re-compile those algorithms and directly reuse the classical recommenders provided by LibRec. There are two main purposes to include those traditional recommendation algorithms – On one hand, those algorithms can be applied after the data transformation (e.g., splitting operations), which is an essential step in the context-aware transformation algorithms. On the other hand, it is usually common to compete a contextual recommendation algorithm with non-contextual algorithms to

\(^{13}\text{Sparse Matrix Storage, http://netlib.org/linalg/html_templates/node90.html}\)
judge whether the contextual effect is significant or a context-aware recommendation algorithm is necessary or not.

4) Configuration and Evaluations: For evaluation purpose, we provide DataSplitter which enables the users to adopt either train-testing evaluation or the $N$-folds cross validations.

Most of the recommendation algorithms embedded in CARSKit are able to perform the two recommendation task: rating prediction and item recommendation, except those ones specifically designed for top-$N$ recommendation, such as CSLIM. But the evaluation is different from traditional ones, since contexts are additional inputs in the evaluation process. Typically, the rating prediction can be evaluated by different prediction errors, such as mean absolute error (MAE), root mean square error (RMSE) and mean prediction error (MPE). The item recommendation can be evaluated through relevance metrics, such as precision and recall, and ranking metrics, such as mean average precision (MAP), normalized discounted cumulative gain (NDCG) and mean reciprocal rank (MMR).

Fig. 2: Sample of Evaluations on DePaulMovie Data

We simply explored a 5-fold cross validation on DePaulMovie data\textsuperscript{14} by using recommendation algorithms built upon matrix factorization. The evaluation results based on rating predictions (by RMSE) and top-5 recommendations (by NDCG) are shown in Figure 2. Among of those algorithms, BiasedMF is a standard matrix factorization technique and also a non-contextual algorithm applied on the context-aware data. UserSplitting and ItemSplitting are two contextual splitting approaches [10] where BiasedMF is applied after data transformation. In addition, deviation-based CAMF (e.g., CAMF_CI and CAMF_CU) [2] and similarity-based CAMF (e.g., CAMF_ICS) [15] are also considered. The results reveal that CAMF_ICS is able to help obtain the best NDCG on this data set, while there are no statistical significant differences on RMSE among those algorithms.

Moreover, CARSKit provides flexible configurations by a single file which includes both algorithm configuration (e.g., algorithm parameters) and experimental configuration, e.g., inputs and output, evaluation strategy and metrics, etc.

IV. CONCLUSIONS AND FUTURE WORK

In this paper, we introduced CARSKit which is an open-source Java-based context-aware recommendation engine. CARSKit contributes to the domain of recommender systems by providing 1) the 1$^{st}$ library specifically designed for context-aware recommendation; 2) implementations of the state-of-the-art contextual recommendation algorithms; 3) efficient data structure and flexible configuration, as well as multi-aspect evaluations. In future, we will add more state-of-the-art context-aware recommendation algorithms to CARSKit. It is also possible to include context suggestion algorithms, where context suggestion or context recommendation [11], [8] is a new direction derived from CARS which aims to recommend appropriate contexts for users to consume the items.

V. ACKNOWLEDGEMENT

We would like to show our gratitude to Dr. Guibing Guo (the author of LibRec) for his comments and suggestions on the development of CARSKit.

REFERENCES


\textsuperscript{14}There are 5,029 ratings (scale 1-5) by 97 users on 79 movies within contexts “time, location, companion”.