An Extended Data Model Format for Composite Recommendation

Alan Said[1], Babak Loni[1], Roberto Turrin[2], Andreas Lommatzsch[3]
[1]TU-Delft, The Netherlands
[2]Moviri, Italy
[3]TU Berlin, Germany
alansaid@acm.org, b.loni@tudelft.nl, roberto.turrin@moviri.com, andreas@dai-lab.de

ABSTRACT

Current *de facto* data model standards in the recommender systems field do not support easy encoding of heterogeneous data aspects such as context, content, social ties, etc. In order to facilitate a simpler means of sharing and using the rich datasets used by research- as well as production systems today, in this paper we propose a data model standard for heterogeneous datasets in the recommender systems domain. The data model is based on the classical tab separated value (TSV) data model with additional fields for encoding relational data in JSON format. Through using already established data sharing formats, we intend to make the usage of the data model as effortless as possible, i.e. there already exist generic tools for parsing and managing the data format in most programming languages. We invite the RecSys community to contribute to the proposed data model in order to increase ease of use and adoption.

Categories and Subject Descriptors

H.3.5 [**Online Information Services**]: Data Sharing

General Terms

Design, Documentation, Standardization

1. INTRODUCTION

Recommender systems research is inherently based on the underlying data available for the recommender systems to use. No matter whether the systems utilize interaction data such as ratings or purchases, or the content of the recommendable items to create recommendations, there is always a dataset which serves as the basis for recommendation.

In recent years, recommender systems research has seen a plethora of recommendation approaches similar to, or based on, the recommender problem defined by the Netflix Prize[1], i.e. predicting the ratings users give to movies. Recommendation of this type is often based simply on the user-item interactions, thus the only data necessary for this purpose is the triple $\{u, i, r_{u,i}\}$, where u is a user in the set of users U, i an item in the set of items I, and $r_{u,i}$ the rating given by user u on item i. Sometimes, this will also include a timestamp that the interaction was created on. This has become the *de facto* standard for data sharing and data handling in the RecSys community, e.g. the recommendation frameworks LensKit, MyMediaLite and Apache Mahout all support this type of data as input, sometimes without other alternatives. Similarly, common recommendation data sets are based on this type of data (e.g. Movielens, LastFM, etc.). Some datasets include additional data – there is however no standard for sharing (or using) it.

In this work, we propose a relational data model for rich, heterogeneous data which can be used for recommendation. The purpose is to facilitate easy sharing of this type of data. Our data model is based on an extended version of the traditional TSV/CSV data model currently used in common recommendation frameworks and datasets.

2. DATA MODEL FORMAT

Our hybrid data model is inspired by *multigraphs*, i.e. it is a set of *nodes* and *edges*. In a multigraph, an edge connects two nodes, and two nodes might be connected by multiple edges. The multigraph data structure allows to easily represent any entity as a node and any edge between entities as a relation. Therefore, in our proposed data model we define the following two concepts:

- **Entities** that correspond to the nodes of the graph. An entity can be a user, an item or any context. An entity can contain optional properties and it can be connected to other entities through a relation.
- **Relations** that correspond to the edges in the graph. A relation connects two entities and contains properties which specify how the entities are connected.

Entities and relations consist of a set of tab-separated fields, Listings 1 and 2 presents examples of this.

Listing 1: Representation of a single Entity

```
etype \t eid \t timestamp \t properties \t
    linked_entities
```

Listing 2: Representation of a single Relation

```
rtype \t rid \t timestamp \t properties \t
    linked_entities
```

2.1 Data Model Implementation

Table 1 presents the datatype and descriptions of each of the fields of entities or relations. Both the entity and relation concepts have the same format, they can however be interpreted differently. The fields *properties* and *link_entities* contain a JSON-encoded column which allows the inclusion of any number of properties (or linked_entities) in key-value pairs. If a property has multiple values, the different values can be represented as a JSON array. We illustrate this in an example in Section 2.2.

Fig. 1 schematizes the relation between a user and a movie seen by the user; the movie is characterized by two linked entities (the genre and the actor) and a property (the title). Note that a relation

[1]http://www.netflixprize.com

RecSys 2014 Poster Proceedings, October 6 – 10, 2014, Foster City, Silicon Valley, USA.

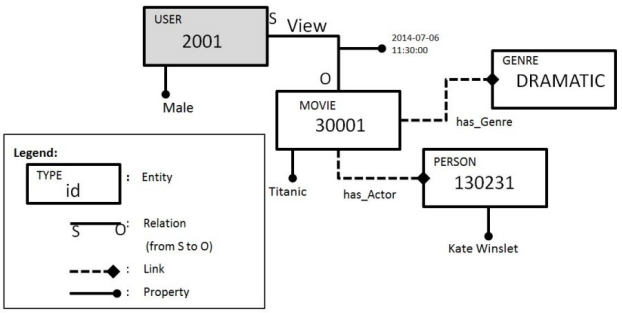

Figure 1: An entity represented in the proposed data model format.

Table 1: Fields of an Entity or a Relation record

Name	Type	Description
etype/rtype	string	representing the type of entity/relation
eid/rid	string	representing the id of the entity/relation
timestamp	long	the Unix epoch time indicating creation time of entity/relation
properties	JSON	a JSON-formatted string indicating the properties of entity/relation
linked_entities	JSON	a JSON-formatted string indicating the linked entities

(solid line) represents a connection that links multiple entities (e.g., the rating given by a user to a movie seen at home with his partner using a smartphone). In Figure 1 the relation 'View' has a subject (S) and an object (O). A relation typically occurs at some point in time (e.g., when a user gives a rating, reads a book, befriends another user, etc.). Conversely, a linked entity represents a fact and connects one main entity to another (e.g., the movie has an actor, where the movie is the main entity).

Different scenarios can be described by means of the proposed data model, as shown in Fig. 2, where we represent four use cases: explicit rating, implicit rating, social connections, and contextual data (device). To understand the data model, below we provide a sample showing how a dataset can be represented based on the proposed data model format.

Listing 3: Representation of a MovieTweetings rating and its corresponding entities with our proposed composite format

```
rating.explicit \t 1001 \t 129121892189 \t {
    rating:5} \t {subject:"user:1002",object:"
    movie:2202"}
user \t 1002 \t 129121892189 \t {twitterId:"
    177651718",gender:"male",city:"Barcelona"}
     \t
movie \t 2202 \t 129121892189 \t {title:"Pulp
     Fiction",year:"1994"} \t {actors:["person
    :3001","person:3004"],director:"person
    :3003"}
person \t 3001 \t \t {gender:"male",name:"
    Travolta, John"} \t
person \t 3004 \t \t {gender:"male",name:"
    Jackson, Samuel"} \t
person \t 3003 \t \t {gender:"male",name="
    Tarantino, Quentin"} \t
```

2.2 MovieTweetings Example

MovieTweetings [1] is a dataset consisting of movie ratings that were contained in tweets. The dataset consists of three files that are formatted similarly to the MovieLens dataset ('::' separated). The

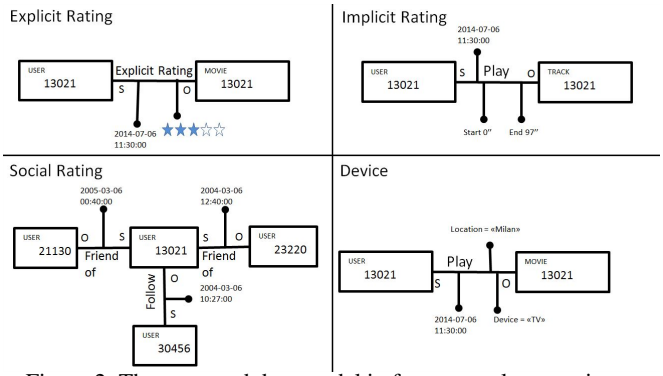

Figure 2: The proposed data model in four example scenarios.

three files *users.dat*, *items.dat* and *ratings.dat* contain information about users, items and ratings accordingly. Formatting this dataset with our proposed format allows to extend the dataset with any potential context and metadata about ratings. A sample representation of a relation and its corresponding entities in our proposed format is shown in Listing 3.

3. DISCUSSION & CONCLUSIONS

The presented data format combines simplicity with a comprehensive descriptive power. We have illustrated that the data model is universally applicable covering explicit/implicit rating scenarios as well as its suitability for describing contexts and user profiles. Inspired by a graph model, the data model is easily extendable and open for integration of additional dataset.

In comparison with XML-based modelling approaches, it minimizes the overhead relying on data formats such as CSV/TSV and JSON. The efficient representation of data makes the data format well-suited for huge sparse datasets typically used in recommendation scenarios. In contrast to data formats distributing information over several different files (e.g., used in the MovieTweeting dataset [1]), the presented data format represents all information in a unified format in one file, simplifying parsing and processing of the data.

In addition, a unified universal data model also helps to overcome the fragmentation of frameworks in the recommendation domain. Since the presented data model covers all aspects relevant for representing data in the recommendation domain, it simplifies the development of recommendation and evaluation frameworks for recommender algorithms using more than the traditional user-item interaction matrix as foundation.

To facilitate the usage and adoption of the data model, we are currently developing open source tools for using this model[2] [2] in combination with common recommendation frameworks. The intention is to create a common set of guidelines for data sharing where the RecSys community is encouraged to actively participate either in the form of development efforts or by proposing changes and additions which should be included in future versions of the data model.

Acknowledgments

This research is supported by funding from the European Commission's 7th Framework Program under grant agreements no. 610594 (CrowdRec).

4. REFERENCES

[1] S. Dooms, T. De Pessemier, and L. Martens, 'Movietweetings: a movie rating dataset collected from twitter', in *CrowdRec Workshop*, (2013).

[2] A. Said, M. Larson, D. Tikk, P. Cremonesi, A. Karatzoglou, F. Hopfgartner, R. Turrin, and J. Geurts, 'User-item reciprocity in recommender Systems:Incentivizing the crowd', in *UMAP ProS Workshop*, (2014).

[2] http://github.com/crowdrec/datasets