**Article Title**

**Authors**

**Affiliations**

**Corresponding author**

**Abstract**

**Keywords**

**Specifications table**

**Value of the data**

**Data description**

The structure of Hudup dataset has 10 main units such as “*config*”, “*account*”, “*attribute\_map*”, “*nominal*”, “*user*”, “*item*”, “*rating*”, “*context\_template*”, “*context*”, and “*sample*”. In other words, Hudup dataset is composed of such ten units. Unit “config” establishes basic configurations over entire dataset in form of key-value pairs.

Hudup dataset is indeed an abstract object, which is instantiated into two forms such as programming object and physical storage. As a programming object, Hudup dataset is modeled as *Dataset* which is retrieved and accessed by programmers. As physical storage, Hudup dataset is stored as directory in file system or database in database management system (DBMS). Of course, there is always interaction between *Dataset* and physical storage, which is dependent on applications and purposes of Hudup framework. Thus, it is possible to identify Hudup dataset with the programming object *Dataset*. For programming language Java, *Dataset* is an interface which is implemented by realized objects. Similarly, each unit is an abstract object too, which is also instantiated into two forms such as programming object and physical storage. Regarding physical storage, unit is stored as CSV file, Excel file, or table for DBMS. However, programming objects for units are more plentiful. For example, “config” is modeled as a map or dictionary whereas “attribute\_map” is modeled by both objects *InternalRecord* and *ExternalRecord*. Units “account”, “user”, and “item” are modeled as *Profile* along with *Attribute*. Unit “nominal” is modeled as *Nominal* object. Unit “rating” is modeled as collections of *RatingVector*. Units “context\_template” and “context” are modeled as *ContextTemplate* and *Contex*t, respectively. Recall that objects *Dataset*, *InternalRecord*, *ExternalRecord*, *Profile*, *Attribute*, *Nominal*, and *RatingVector* are programming objects which provide properties and facilitated methods for programmers to process and access units. Obviously, each unit can be accessed by some specified objects in powerful manner, but *Profile* object can model all objects. In other words, all units can be accessed by Profile object which is the most flexible object. Please refer to the book chapter “Hudup: A Framework of E-commercial Recommendation Algorithms” (Nguyen & Do, 2015) to know some of these objects. Of course, *Dataset* has many methods to retrieve all other objects. In general, these objects are also abstract. However, Hudup dataset and units has the top-most abstract level.

In physical storage system, units are stored as CSV files, Excel files or database tables in form of tables whose columns are fields or attributes. Following table lists fields for each unit.

|  |  |
| --- | --- |
| **Table** | **Fields** |
| “config” | attribute, attribute\_value |
| “account” | account\_name, account\_password, account\_privs |
| “attribute\_map” | internal\_unit, internal\_attribute\_name, internal\_attribute\_value,  external\_unit, external\_attribute\_name, external\_attribute\_value |
| “nominal” | nominal\_ref\_unit, attribute, nominal\_index, nominal\_value, nominal\_parent\_index |
| “user” | userid, user\_type, field1, field2, etc. |
| “item” | itemid, item\_type, field1, field2, etc. |
| “rating” | userid, itemid, rating, rating\_date |
| “context\_template” | ctx\_templateid, ctx\_name, ctx\_type, ctx\_parent |
| “context” | userid, itemid, ctx\_templateid, ctx\_value, rating\_date |
| “sample” | sample\_field1, sample\_field2, sample\_field3, etc. |

**Experimental design, materials, and methods**

**Acknowledgements**

**Competing interests**

**References**