Recommendation algorithms with maximum prediction quality

Scenario

Recommendation engines (REs) are increasingly being used in e-commerce for product recommendations. Recommendation algorithms calculate and automatically recommend products on the basis of product detail views opened by visitors to web shops. This maximises the user's activity (number of views opened) and the success (sales, turnover). Developing powerful algorithms for REs is currently one of the most popular areas of research focus in data mining.

In the scenario in question, the operator of a web shop would like to use a recommendation engine, which maximises both activity and success, with success being weighted higher. It is, therefore, a matter of selecting the best algorithm.

Three types of transaction are considered for each web session: opening a product detail view, placing a product in the shopping basket and purchasing a product. A session typically takes the following course: the user browses in the web shop, opening product detail views as he goes. If the user likes a product, he will place it directly in his shopping basket. At the end of the session, the user can then click on his basket and order the products he is interested in.

Tasks

The DMC 2011 competition consists of two tasks. These are assessed independently of each other.
The first task involves statically analysing an algorithm i.e. training the algorithm to historical transaction data, the training data. In order to be able to evaluate the prediction quality of the recommendations, the first transactions are specified on a test quantity for each session. These are the test data. The objective of the algorithm is to predict the remaining transaction data for the session. The generated prediction file is sent to the prudsys DMC team. The predicted products are then compared against the actual remaining transaction data from the sessions, the evaluation data. The team with the highest score based on the evaluation data wins.

The second task involves dynamically evaluating an algorithm, the implementation of which is sent to the prudsys DMC team. The objective is to apply the algorithm step-by-step to historical transaction data and continuously predict the next products in a session. As it receives all the transactions from each session one after the other in succession, it learns and predicts at the same time. The team with the highest score over all prediction steps wins.

Data

Data files are required for the individual tasks. These data files are structured text files containing data sets. Below are some points to note about the files:

1. Each data set is in a row of its own, ending with „CR“ („carriage return“, 0xD) or „CR“ and „LF“ („carriage return“ and „line feed“, 0xD and 0xA).
2. The first row has the same structure as the data sets, but contains the names of the respective columns (data fields).
3. The top row and each data set contain several fields separated from each other by the „|“ symbol.
4. There is no escape character, quotes are not used.
5. ASCII is the character set used.

The only column headings that can actually appear here are SessionNo, ItemNo and TransactType. The field values contain character strings. The SessionNo and ItemNo columns can contain any nonnegative, integral values, whereas the range of values for TransactType is limited to 0, 1 and 2.

The training file for the DMC has the following format:

Column name	Description
SessionNo	Consecutive number of sessions
ItemNo	Number of product
TransactType	Transaction type (0 – viewed, 1 – placed in basket, 2 – purchased)

The data corresponds to real anonymised shop data. Duplicate rows and sessions with fewer than 2 product views have been omitted. The data sets were further anonymised by deleting, adding and modifying transactions from the remaining sessions. When the data was being prepared, it was ensured that products were always placed in the basket before they were purchased. Similarly, products were always viewed immediately before they were placed in the basket. Products can be viewed several times within one session.

Training data is supplied in the data file *transact_train.txt* . The example below illustrates a fictitious excerpt from the file:

```
1000|133|0
1000|444|0
1000|444|1
1000|15|0
1000|17|0
1000|17|1
1000|444|2
1001|333|0
1001|133|0
1001|133|1
```

In the 1000th web session the user first looks at product 133, then product 444, which he then places in his basket, then looks at product 15, then product 17, which he then also adds to his basket and finally orders product 444. In the 1001st web session the user looks at products 333 and 133 one after the other and then adds the latter to his basket.

The training file *transact_train.txt* can be used by participants to learn a recommendation model for task 1 and to develop a powerful online algorithm for task 2. Further data is explained in the tasks.

Entries

Participants can submit their results up until 31st May 2011, 24:00 (UTC+2, or CEST). The tasks below explain how to submit entries.

Task 1: Static learning

Data

In the test data set - unlike in the training data set - products occurring more than once per session are replaced by their highest transaction type. In other words, if a product has been viewed several times, only the first viewing will be retained. If a product has been placed in the basket, the first viewing is labelled basket; if the product has been purchased, the viewing is labelled purchase. This means that products cannot appear more than once in one session.

The resulting data is now divided into two files: the first file *transact_test.txt* contains all the transactions in the session, except for the last 3. These remaining transactions are saved in the second file *transact_eval.txt* (evaluation file).

For the example above, this produces the following entries in*transact_test.txt*:

```
1000|133|0
```

Similarly, the entries in*transact_eval.txt are:*

```
1000|444|2
1000|15|0
1000|17|1
```

It is important to note that the 1001st web session does not feature at all, as it only contains two products. At least 4 products are, however, required (as in the 1000th web session), so that the file*transact_test.txt* contains at least one transaction.

Participants receive the *transact_test.txt* file and use it to predict the remaining transactions in the sessions and then send the results as a text file to the prudsys DMC team.

Entries

The results file must have the following format:

Column name	Description
SessionNo	Consecutive number of session ID
ItemNo	Number of product predicted to be purchased

The participants specify for each session the numbers of the 3 products they think will come up in the rest of the session. The order of the predicted products for a session does not matter. The file structure must follow the conventions for data files set out in the introduction.

The excerpt from a results file for the example shown might, therefore, look like this:

```
1000|555
1000|17
1000|15
```

The results file must be sent to dmc_task1@prudsys.de as a zipped text file. The name of both the zip file and the included text file must be made up of the team name, the task number *task1* and the type (zip or txt):

“<Teamname>_task1.zip“, (e.g. *TU_Chemnitz_1_task1.zip*)

i.e.

“<Teamname>_task1.txt“, (e.g. *TU_Chemnitz_1_task1.txt*).

The team name will have been sent to the team leader in the registration confirmation.

Evaluation

For evaluation purposes, the score is calculated on the basis of the evaluation file *transact_eval.txt* as follows: For the *i*- st/th session the score s_i is defined as

$$s_i = n_i^{Clicks} + 5 \cdot n_i^{Baskets} + 10 \cdot n_i^{Orders} \quad . \tag{1}$$

Here n_i^{Clicks} equals the number of correctly predicted product views, $n_i^{Baskets}$ is the number of correctly predicted products which were also placed in the basket, and n_i^{Orders} is the number of correctly predicted products which were also purchased.

The results are, therefore, compared with the actual transactions in the evaluation file. In the example, products 15 and 17 were correctly predicted in the 1000th session. In addition, product 17 was added to the basket, which further increases the score. The prediction for product 555, however, was incorrect; 444 would have been right. On this basis the score is therefore

$$s_{1000} = 2 + 5 \cdot 1 + 10 \cdot 0 = 7 \, .$$

By applying the scoring to all *n* sessions the overall score *s* is the total of all the session scores:

$$s = \sum_{i=1}^{n} s_i \tag{2}$$

This is calculated for each entry. Entries with the highest overall scores win. *(In the event of a tie, lots shall be drawn)*

Task 2: Dynamic learning

Data

Transaction data is not issued for this task. Participants may, however, use the training data file to test their own implementation: *transact_train.txt.* The data on which the application is then ultimately evaluated, has the same structure as the data in *transact_train.txt*. Once again, the data here was anonymised, however, this data is from a different shop.

Parameters for implementation

Implementation must take the form of a Java class. It must be able to run on a PC with 1024 MB guaranteed heap space and a 1.6 GHz Intel Core Duo Processor. The "sandpit" principle applies, i.e. the application must not connect to external resources (e.g. http connections). Java libraries may be used, but these must be under free licence. The application must not create files, incur expenditures or emit signals.

Furthermore, the class must implement the communications interface listed below *RealtimeRecommInterface*, which in turn defines two methods via which the application can be activated. In keeping with the nature of an online shop, the response times of the functions in the technical conditions specified above must not exceed 500 milliseconds. The class must be initialised via the class's default constructor (no reference parameter). The *TopsellerAgent.java* file contains a sample implementation. This simple implementation includes, in particular, product purchases in the *submitTransaction* method and always recommends products with the most sales independently of the current product view using the *getRecommendations* method.

```
import java.util.List;

public Interface RealtimeRecommInterface {
  /**
   * Returns the recommendations for an item that shall be bought.
   * The recommendations may not contain the requested item.
   * This is a pure application method.
   *
   * @param sessionID the current session ID
   * @param itemID the requested item ID
   * @param maxNumberOfRecommendations maximum number of recommendations
   * @return vector of recommended item IDs
   * @throws Exception
   */
  public List<String> getRecommendations(String sessionID,
    String itemID, int maxNumberOfRecommendations) throws Exception;

  /**
   * Submits a transaction of an item. This is a pure learning method.
   *
   * @param sessionID the current session ID
   * @param itemID the item ID of the transaction
   * @param transactType the transaction type
   *   (0 – viewed, 1 - added to basket, 2 - ordered)
   * @throws Exception
   */
  public void submitTransaction(String sessionID, String itemID,
    int transactType) throws Exception;
}
```

Entries

Results files must be zipped and sent as one file to dmc_task2@prudsys.de. The zip file must not be larger than 5 MB. The file name must be made up of the team name, the task number *task2* and the type (.zip):

"<Teamname>_task2.zip", (e.g. *TU_Chemnitz_1_task2.zip*).

The team name will have been sent to the team leader in the registration confirmation.

You MUST also state the full name of the class implementing the interface in the email. Any external Java libraries used must also be submitted. The implementation can be submitted as source code or fully compiled bytecode (jar file). If source code is submitted, it must be possible to compile it with a current jdk1.6 version and an Ant buildfile must be included. If bytecode is submitted, it must be executable with a current jre6 version. A short text file outlining the details of the logic of the implemented algorithm would be helpful.

Evaluation

The prudsys DMC team's simulation software imports the transaction file for the online test and invokes the *submitTransaction* method for each transaction. For product views, the *getRecommendations* method is also invoked for each transaction with a maximum number of 3 recommendations. If one of the products recommended via this function is viewed in the very next transaction of the session, the tally of correctly predicted product views will be increased. If, later on in the session, the product is placed in the basket or purchased, the tallies for these respective transactions will be increased in the same way. Please note that products are not allowed to recommend themselves. At the end of the session the score will be calculated as per formula (1). The overall score for all sessions is calculated using formula (2). The submitted application should be able to learn online and continuously increase its prediction quality.

The invocations for the sample sessions from task 1 look like this:

```
// Session 1000
submitTransaction(1000,133,0); // product 133 has been viewed
recs = getRecommendations(1000,133,3); // get recommendations for
                                       // product 133
submitTransaction(1000,444,0);
recs = getRecommendations(1000,444,3);
```

```
submitTransaction(1000,444,1); // product 444 has been added to the
                               // basket
submitTransaction(1000,15,0);
recs = getRecommendations(1000,15,3);
submitTransaction(1000,17,0);
recs = getRecommendations(1000,17,3);
submitTransaction(1000,17,1);
submitTransaction(1000,444,2); // product 444 has been purchased

// Session 1001
submitTransaction(1001,333,0);
recs = getRecommendations(1001,333,3);
submitTransaction(1001,133,0);
recs = getRecommendations(1001,133,3);
submitTransaction(1001,133,1);
```

For example, if product 444 was recommended for product 133 in the 1000th session, this would be evaluated as a correct basket and purchase recommendation as well as a correct product view recommendation. If product 17 was recommended for product 15, this would be evaluated as a correct product view and basket recommendation. The score for this session with these two recommendations would be $s_{1000} = 2 + 5 \cdot 2 + 10 = 22$.