Information Retrieval and Recommender Systems
Assignment #2 – Recsys Challenge Report

Nancy Yacovzada	Michal Yaacov	Shay Rozen	Ofer Litver
300649894	038178950	204621643	039176052

1. Problem Formulation and First Insights

The problem is formulated as follows: We are presented with datasets captured from an e-commerce Internet site. Given a pair of datasets, describing clicking transactions and actual purchases of products, respectively, we are asked to predict actual purchases from a third datasets referring to another click data.

1.1 Problem Components

More specifically, the components of the problem are:

- `yoochoose-clicks.dat` – a file containing 33,003,944 samples of clicks on products that were recorded. Each row describes a single click and has the following attributes:
    - Session ID – Unique ID of the unique session (out of 9,249,729 unique sessions) during which the click was made. It is possible (and very likely) that during each session more than one click will be made.
    - Timestamp – Field recording the date and time the click was created at. Constructed in the format: `YYYY-MM-DDTHH:MM:SS.SSSZ`. It is unknown to us from which timezone the Timestamp is taken.
    - Item ID – Unique identifier of an item in the e-commerce site.
    - Category – The category into which the item belongs, according to the site's categorization mechanism.
- `yoochoose-buys.dat` – a file containing 1,150,753 samples of purchases of products that were recorded. Each row describes a single purchase and has the following attributes:
    - Session ID – Same as in the "`yoochoose-clicks`" file. It is the same ID numbering that was used in that file, except that now we have only 509,696 unique sessions.
    - Timestamp – Same as in the "`yoochoose-clicks`" file.
    - Item ID – Same as in the "`yoochoose-clicks`" file.
    - Item Price – The price that was paid for that item.

- Item Quantity – Number of units that were purchased from that item.

1.2 Scoring method

1.2.1 Formulation of Scoring

The scoring of a given solution is calculated using the measure:

$$Score(\boldsymbol{Sl}) = \sum_{\forall s \in \boldsymbol{Sl}} \begin{cases} if\ s \in \boldsymbol{Sb} \rightarrow \frac{|S_b|}{|S|} + \frac{A_s \cap B_s}{A_s \cup B_s} \\ else \rightarrow -\frac{|S_b|}{|S|} \end{cases}$$

Where:

$Sl - Sessions\ in\ submitted\ solution\ file$

$S - All\ sessions\ in\ the\ test\ set$

$s - Session\ in\ the\ test\ set$

$S_b - \ Sessions\ in\ test\ set\ which\ end\ with\ buy$

$A_s - Predicted\ bought\ items\ in\ session\ s$

$B_s - Actual\ bought\ items\ in\ session\ s$

It is very clear from the scoring method that Type II errors are more unfavorable than Type I errors. In other words, it is better to miss a session that will eventually buy, rather than recommending for session that will prove not to end with buy.

2. Workflow

2.1 Phase 1 – Data Preparation

In this phase we prepare the data for the process. The following stages are accomplished:

- Loading the data files – the data files were loaded into Numpy record array.
- Preprocessing – feature columns were extracted and appended into the data files (full description is available in appendix 4.2).
- Creating the true label – assigns the relevant labels ("bought"=1, "not bought"=0) for each session derived from the source files.
- Aggregative feature engineering – the list of features detailed in appendix 4.2. After the features generation we performed scaling step which adjusted to each feature as per its characteristics (values scaled to [0,1] range). We used the MinMax scaling and a variation of log transformation for this task.

2.2 Phase 2 – Classification of Sessions

Re-sampling – Due to the large size of the files and our lack of access to descent processing power, we used a more efficient approach for model training; we created a training set consisting of the two classes in a 1:1 ratio. Sooner we found out that this approach wasn't a mere complexity compromise, it also led us to more accurate results as the classifier learned better the patterns of class 1 (sessions that end with buy).

Fitting a classifier – We evaluated few of the most popular classification algorithms, such-as SVM, Decision Tree, Random Forest, KNN and Naïve Bayes. Each algorithm was tested with 5-fold cross validation, using a variety of configurations. Eventually Naïve Bayes was chosen, as it proved to be superior to the rest.

Prediction – Using the Naïve Bayes classifier to make the actual prediction about which of the test file sessions will end with buy.

2.3 Phase 3 – Item Recommendations

In this section we recommend the items for each of the sessions we predicted will end with buy on the previous phase.

Our initial approach was to use well known recommender methods implemented by the Graphlab python package. We tried several algorithms, such as: item-based collaborative filtering, ranking factorization and popular items, but all gave poor results, even negative values. The main issue was that the algorithms predicted the ranks of items which weren't viewed during the sessions, while the probability that a user will buy an item without watching it first is pretty low. Hence we configured the algorithms to include the viewed items in their predictions (exclude_known=False), but the results did not significantly improved, because we still allowed recommending new items.

Later we turned to a more naïve approach – in each session we'd simply recommend the user (session) the top k items he had watched the most. The underlying assumption was that the more a user clicks on item the greater the potential the user will buy this item. This technique performed much better (submission score of 23,000) and supported our intuition – the recommended items should be from the ones which already viewed during the session.

Eventually, we improved the recommender performance by referring the percentage of time spent viewing each item in the session. The underlying assumption was that the longer the time a user watching an item the greater the potential the user will buy this item. We defined 2 conditions: 1) The item time spent percentage is greater than 1%; 2) The max items number recommended is 5. The recommender output must meet at least one of the above

conditions for every session. This technique performed the best with a submission score of 27,430.

3. Conclusions

As mentioned earlier, the challenge included 2 main issues – a classification task and a recommender task. We found that the simple naïve approach is the best strategy in this case. Our intuition that the time spent watching each item in session is correlated with the actual purchased items turned out to be valid.

4. Appendices

4.1 Submissions login data

```
Username: michal

Password: 251g3f
```

4.2 Session Features Used

- TotalTrx – number of transaction in session.
- Week – number of week in year.
- DayInWeek – weekday – represented as 7 Boolean features (isSunday, isMonday, etc.).
- Hour – 24 bins of whole hours
- PartOfDay – part of day (morning/noon/evening/night), represented as 4 Boolean features (isMorning etc.).
- SessionDuration – duration of the entire session, from first to last transaction.
- NumberDistinctCategories – number of categories viewed in the session.
- AvgTimeSpentItemView – average viewing time spent for each item in the session.
- StdTimeSpentItemView – the standard deviation of viewing time spent for each item in the session.
- AvgDoubleClicks – the average of times viewing item in session.
- MaxDoubleClicks – the number of clicks for the item in the session that has been clicked the most times.
- TimeSpentList – a list of the items and the times spent on each item in the session.
- IsBought – the classification target variable (label).

4.3 Existing Packages Being Used

We used the following popular Python packages:

- Numpy
- Scipy
- Sklearn
- Graphlab / Dato – as described above, eventually those packages were not used.