TDT 4215 – Web Intelligence Course

Group Project: Recommender System Using Adressa Dataset

Özlem Özgöbek, Lemei Zhang, Peng Liu

Goal and description of the group project

The task of the group project is to develop a news recommender system. Each group can choose their own tools and develop their own methods in order to get the best results. This document provides detailed explanations and some examples we provide for you. More information about the group project and deadlines is provided in a separate document (Web Intelligence Group Project_2018.pdf).

For the example recommender system, a Python library called scikit-learn library and another scikit called Surprise is used. Detailed documentation about scikit-learn is available at: http://scikit-learn.org/ And for Surprise is available at: surprise.readthedocs.io/en/stable/ Both Surprise and scikit-learn are open source projects where you can download the source code and add your own codes if you want to it for your project.

Remember that this is just a very simple example and you are **free to use** any other programming language, libraries and/or tools to build your own recommender system.

Dataset

The Adressa dataset is a dataset published by the SmartMedia group at NTNU in partnership with the local newspaper Adresseavisen in Trondheim.
The dataset includes the anonymized user data from the digital newspaper. The user data includes more than 30 attributes including the location, timestamp, active time (or reading time), keywords, title and URL of the read article. Adresseavisen has an online user subscription system. Subscribed users provide longer terms of consistent data, so it might be easier to develop a recommender system only for the subscribed users. Unsubscribed users can only be tracked within a session. For more details about the dataset please check the related documentation (cxdataset_v3.0.pdf) and the processing the data section below. Remember that this is a real world data so inconsistencies and noise is available.

The released dataset has two versions in different sizes. The lighter version contains data from one week of data collection and it has a compressed size of 1.4 GB. The full version of the dataset contains data from 10 weeks of data collection and has the compressed size of 16 GB. For your group project you can use any of these two datasets, but we recommend you to use the lighter dataset for the ease of processing.

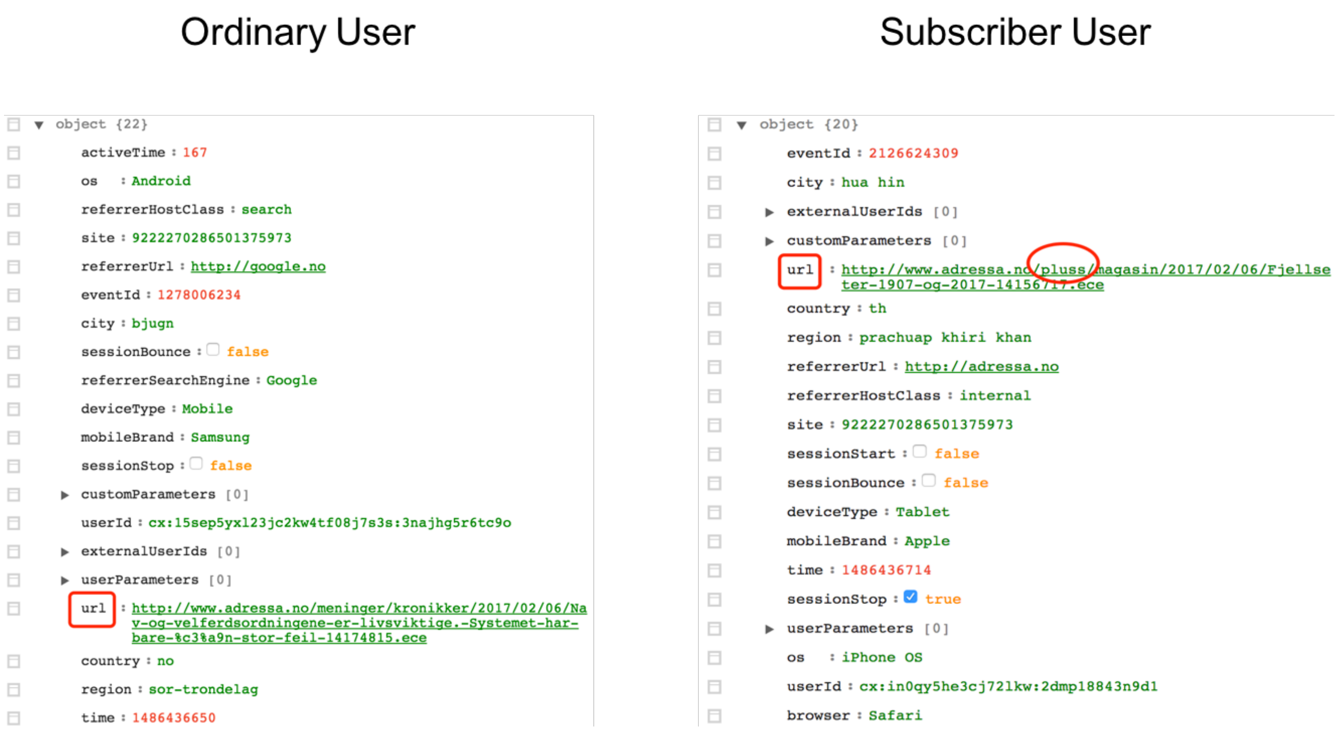
The datasets and the basic documentation for the lighter dataset is available at:
http://reclab.idi.ntnu.no/dataset

Full articles (in Norwegian) from the newspaper can also be provided upon your request if you would like to use more details about the text in your recommender system. For this, text processing and the use of natural language processing tools are required.

Processing the Data

The dataset contains several attributes. Since the dataset contains real world data, there are inconsistencies and noise in the data. For example, not all the attributes are available for all users.

Adressa dataset contains two types of users: Subscribed and unsubscribed users. Subscribed users are logged in while they are browsing and reading the online newspaper. Only subscribed users have the right to access the content behind the paywall, in Adresseavisen's case, this content is PLUSS articles. If you want to differentiate the subscribed users, you can filter users who read the "pluss" articles. In the dataset you should look for url attributes or canonicalUrl attributes contains "pluss" (Figure 1).


Ordinary User

```
▼ object {22}
    activeTime : 167
    os    : Android
    referrerHostClass : search
    site : 9222270286501375973
    referrerUrl : http://google.no
    eventId : 1278006234
    city : bjugn
    sessionBounce : ☐ false
    referrerSearchEngine : Google
    deviceType : Mobile
    mobileBrand : Samsung
    sessionStop : ☐ false
  ▶ customParameters [0]
    userId : cx:15sep5yxl23jc2kw4tf08j7s3s:3najhg5r6tc9o
  ▶ externalUserIds [0]
  ▶ userParameters [0]
    url : http://www.adressa.no/meninger/kronikker/2017/02/06/Na
          v-og-velferdsordningene-er-livsviktige.-Systemet-har-
          bare-%c3%a9n-stor-feil-14174815.ece
    country : no
    region : sor-trondelag
    time : 1486436650
```

Subscriber User

```
▼ object {20}
    eventId : 2126624309
    city : hua hin
  ▶ externalUserIds [0]
  ▶ customParameters [0]
    url : http://www.adressa.no/pluss/magasin/2017/02/06/Fjellse
          ter-1907-og-2017-14156717.ece
    country : th
    region : prachuap khiri khan
    referrerUrl : http://adressa.no
    referrerHostClass : internal
    site : 9222270286501375973
    sessionStart : ☐ false
    sessionBounce : ☐ false
    deviceType : Tablet
    mobileBrand : Apple
    time : 1486436714
    sessionStop : ☑ true
  ▶ userParameters [0]
    os    : iPhone OS
    userId : cx:in0qy5he3cj721kw:2dmp18843n9d1
    browser : Safari
```

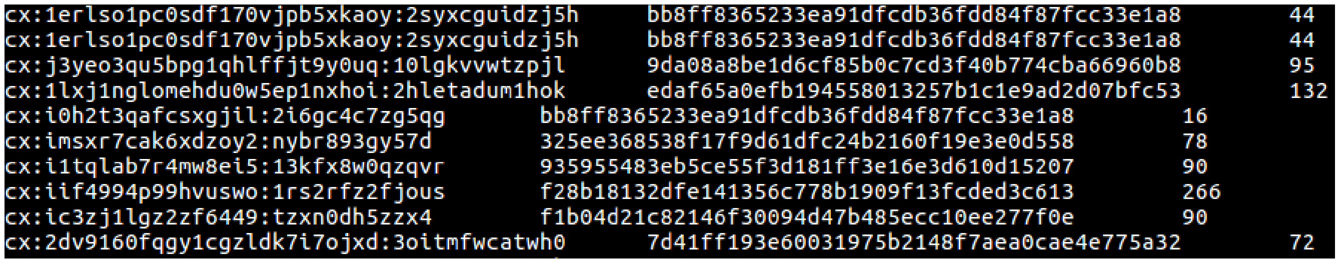
Figure 1. Differentiating the subscribed users.

Missing Attributes: Some events in the dataset don't contain some attributes e.g "keywords"/"profile" because they do not appear in the original dataset. This can be because either the articles do not contain such attributes or they are not types of news articles.

For more details about the dataset please refer to the dataset documentation (cxdataset_v3.0.pdf).

Preprocessing the data: In order to develop your recommender system you should choose a recommendation strategy and decide which attributes of the dataset to use. Before starting to work on the recommender system, you may want to filter out the unnecessary attributes or clean the dataset a little bit. This would make it easier to work on the dataset.

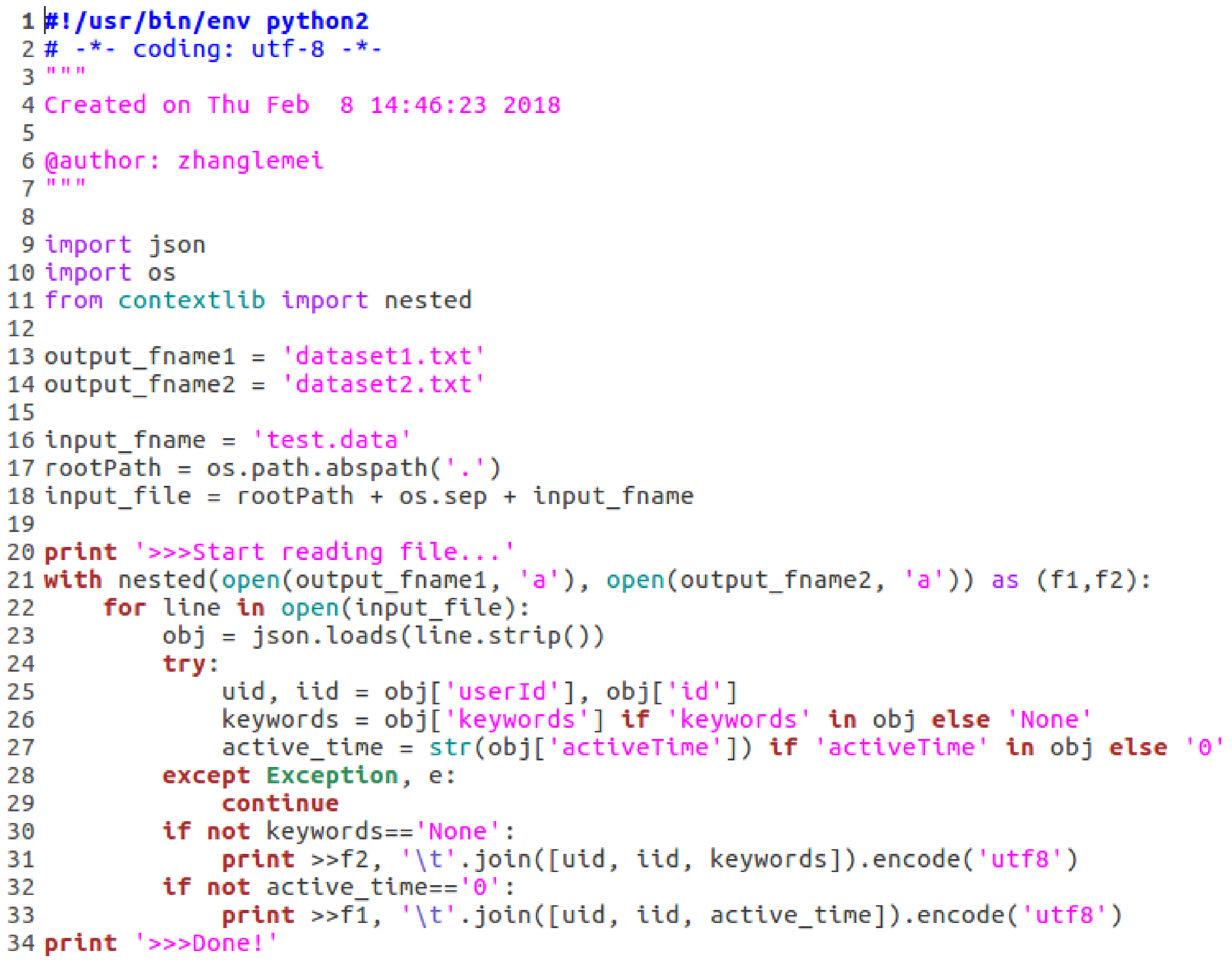
As an example we have filtered only 3 attributes (user id, item id, active time) (Figure 2):



```
cx:1erlso1pc0sdf170vjpb5xkaoy:2syxcguidzj5h     bb8ff8365233ea91dfcdb36fdd84f87fcc33e1a8        44
cx:1erlso1pc0sdf170vjpb5xkaoy:2syxcguidzj5h     bb8ff8365233ea91dfcdb36fdd84f87fcc33e1a8        44
cx:j3yeo3qu5bpg1qhlffjt9y0uq:10lgkvvwtzpjl      9da08a8be1d6cf85b0c7cd3f40b774cba66960b8        95
cx:1lxj1nglomehdu0w5ep1nxhoi:2hletadum1hok      edaf65a0efb194558013257b1c1e9ad2d07bfc53        132
cx:i0h2t3qafcsxgjil:2i6gc4c7zg5qg       bb8ff8365233ea91dfcdb36fdd84f87fcc33e1a8        16
cx:imsxr7cak6xdzoy2:nybr893gy57d        325ee368538f17f9d61dfc24b2160f19e3e0d558        78
cx:i1tqlab7r4mw8ei5:13kfx8w0qzqvr       935955483eb5ce55f3d181ff3e16e3d610d15207        90
cx:iif4994p99hvuswo:1rs2rfz2fjous       f28b18132dfe141356c778b1909f13fcded3c613        266
cx:ic3zj1lgz2zf6449:tzxn0dh5zzx4        f1b04d21c82146f30094d47b485ecc10ee277f0e        90
cx:2dv9160fqgy1cgzldk7i7ojxd:3oitmfwcatwh0      7d41ff193e60031975b2148f7aea0cae4e775a32        72
```

Figure 2. An example of the filtered dataset. Only three attributes are included (user id, item id, active time)

In the dataset, active time represents the seconds that a user spent on an article (e.g. user A spent X seconds on article B). In our example, active time is used as a rating. The code we have used to filter these attributes is shown in Figure 3.

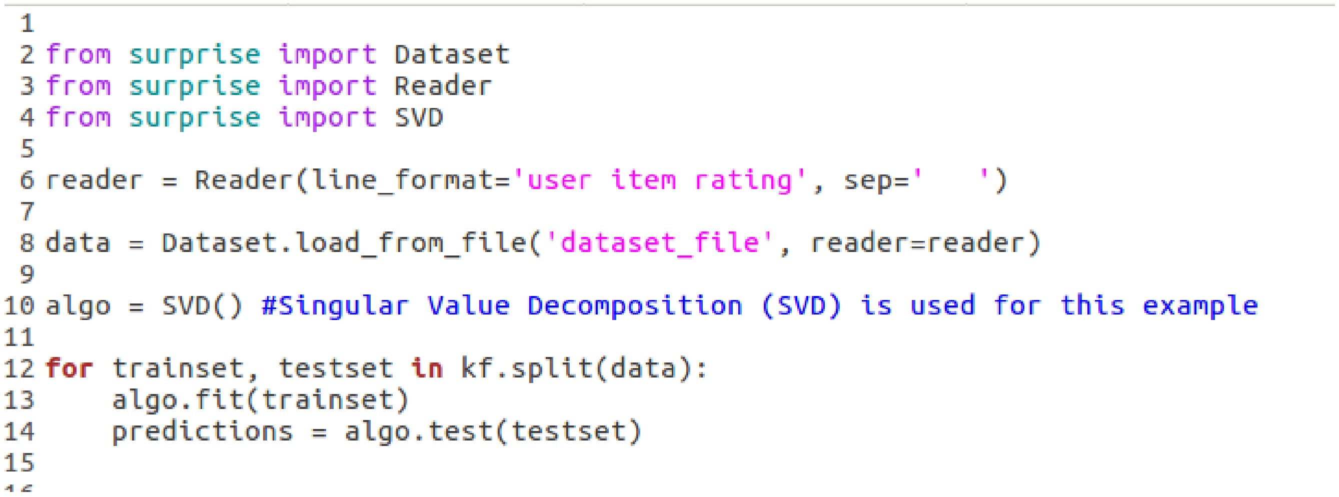


```python
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Thu Feb  8 14:46:23 2018

@author: zhanglemei
"""

import json
import os
from contextlib import nested

output_fname1 = 'dataset1.txt'
output_fname2 = 'dataset2.txt'

input_fname = 'test.data'
rootPath = os.path.abspath('.')
input_file = rootPath + os.sep + input_fname

print '>>>Start reading file...'
with nested(open(output_fname1, 'a'), open(output_fname2, 'a')) as (f1,f2):
    for line in open(input_file):
        obj = json.loads(line.strip())
        try:
            uid, iid = obj['userId'], obj['id']
            keywords = obj['keywords'] if 'keywords' in obj else 'None'
            active_time = str(obj['activeTime']) if 'activeTime' in obj else '0'
        except Exception, e:
            continue
        if not keywords=='None':
            print >>f2, '\t'.join([uid, iid, keywords]).encode('utf8')
        if not active_time=='0':
            print >>f1, '\t'.join([uid, iid, active_time]).encode('utf8')
print '>>>Done!'
```

Figure 3. Python code we have used to filter attributes.

Splitting the dataset: In order to obtain test and training datasets you need to split your dataset. This can be done in various ratios, e.g. %20 to %80. We will provide a separate test dataset for comparable evaluations which is separated from the provided dataset.

A simple example of how to split the dataset by using Surprise (Figure 4):



```

from surprise import Dataset
from surprise import Reader
from surprise import SVD

reader = Reader(line_format='user item rating', sep='    ')

data = Dataset.load_from_file('dataset_file', reader=reader)

algo = SVD() #Singular Value Decomposition (SVD) is used for this example

for trainset, testset in kf.split(data):
    algo.fit(trainset)
    predictions = algo.test(testset)

```

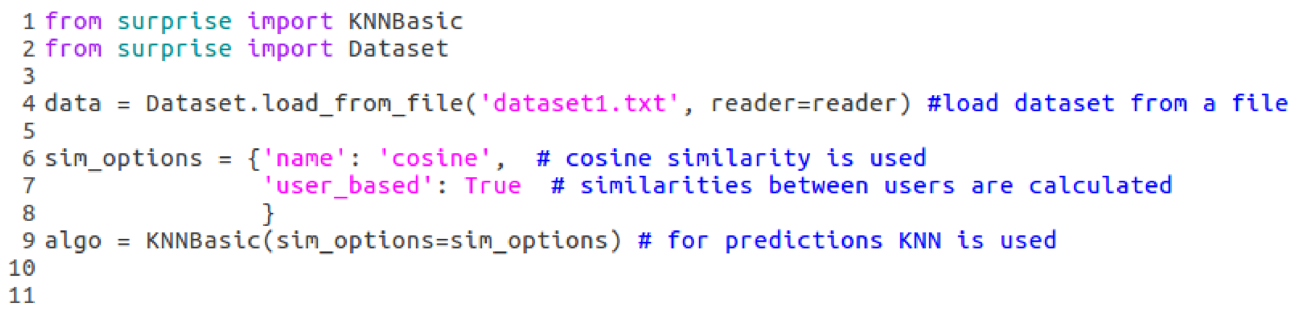
Figure 4. Splitting dataset by using Surprise.

Recommender System

As mentioned before, you are free to choose the recommendation methods, programming languages and tools for your own recommender system. We have prepared a simple example in order to help to get started. Our example includes only collaborative filtering and content based filtering.

Collaborative Filtering

For collaborative filtering we need to find the similarities between users (for user based collaborative filtering) or items (for item based collaborative filtering). There are several similarity measures that can be used. In Surprise some of these similarity measures are implemented and it is quite easy to use:



```
from surprise import KNNBasic
from surprise import Dataset

data = Dataset.load_from_file('dataset1.txt', reader=reader) #load dataset from a file

sim_options = {'name': 'cosine',  # cosine similarity is used
               'user_based': True  # similarities between users are calculated
               }
algo = KNNBasic(sim_options=sim_options) # for predictions KNN is used


```

Figure 5. A simple collaborative filtering example by using Surprise.

Surprise is implemented by partially using scikit-learn, so it is recommended to check the source codes in order to learn and/or improve the implementation.

Content-based Filtering

For a content-based filtering example we have used the TF-IDF (Term Frequency - Inverse Document Frequency). It is possible to implement TF-IDF in Python by using scikit-learn.

TfidfTransformer in scikit-learn transforms a count matrix to a TF-IDF representation:
http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html#sklearn.feature_extraction.text.TfidfTransformer

To get a count matrix from a text, it is possible to use CountVectorizer module in scikit-learn:
http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

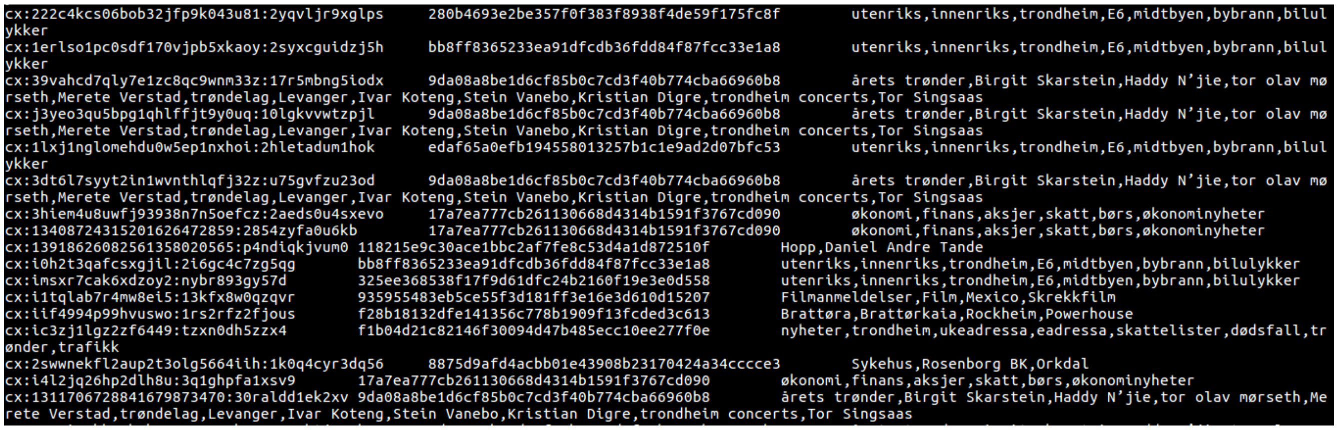
As a combination of the two modules above TfidfVectorizer module can be used directly, especially with raw documents:
http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

These two modules can also be used with the full news articles.

Remember that not all events in the dataset contains keywords, if you would like to use the keywords for TF-IDF you can filter out the events that only include keywords as it is shown in Figure 3.

In a content-based recommender system, after we get the TF-IDF representation for each article in the dataset, we need to compute the recommendations based on the users' reading behaviour and the article similarities. As mentioned in the dataset section above, user-id of unsubscribed users are based on the session so it is not possible to track a user's reading behaviour between sessions. Besides, one session for one user may include only a few articles which makes it difficult to get good recommendation results.

In Figure 6, it is shown the filtered dataset for content-based filtering. The first column represents the user-id, second column represents the item-id and the last column includes the keywords for each article.



```
cx:222c4kcs06bob32jfp9k043u81:2yqvljr9xglps     280b4693e2be357f0f383f8938f4de59f175fc8f        utenriks,innenriks,trondheim,E6,midtbyen,bybrann,bilul
ykker
cx:1erlso1pc0sdf170vjpb5xkaoy:2syxcguidzj5h     bb8ff8365233ea91dfcdb36fdd84f87fcc33e1a8        utenriks,innenriks,trondheim,E6,midtbyen,bybrann,bilul
ykker
cx:39vahcd7qly7e1zc8qc9wnm33z:17r5mbng5iodx     9da08a8be1d6cf85b0c7cd3f40b774cba66960b8        årets trønder,Birgit Skarstein,Haddy N'jie,tor olav mø
rseth,Merete Verstad,trøndelag,Levanger,Ivar Koteng,Stein Vanebo,Kristian Digre,trondheim concerts,Tor Singsaas
cx:j3yeo3qu5bpg1qhlffjt9y0uq:10lgkvvwtzpjl      9da08a8be1d6cf85b0c7cd3f40b774cba66960b8        årets trønder,Birgit Skarstein,Haddy N'jie,tor olav mø
rseth,Merete Verstad,trøndelag,Levanger,Ivar Koteng,Stein Vanebo,Kristian Digre,trondheim concerts,Tor Singsaas
cx:1lxj1nglomehdu0w5ep1nxhoi:2hletadum1hok      edaf65a0efb194558013257b1c1e9ad2d07bfc53        utenriks,innenriks,trondheim,E6,midtbyen,bybrann,bilul
ykker
cx:3dt6l7syyt2in1wvnthlqfj32z:u75gvfzu23od      9da08a8be1d6cf85b0c7cd3f40b774cba66960b8        årets trønder,Birgit Skarstein,Haddy N'jie,tor olav mø
rseth,Merete Verstad,trøndelag,Levanger,Ivar Koteng,Stein Vanebo,Kristian Digre,trondheim concerts,Tor Singsaas
cx:3hiem4u8uwfj93938n7n5oefcz:2aeds0u4sxevo     17a7ea777cb261130668d4314b1591f3767cd090        økonomi,finans,aksjer,skatt,børs,økonominyheter
cx:13408724315201626472859:2854zyfa0u6kb        17a7ea777cb261130668d4314b1591f3767cd090        økonomi,finans,aksjer,skatt,børs,økonominyheter
cx:13918626082561358020565:p4ndiqkjvum0 118215e9c30ace1bbc2af7fe8c53d4a1d872510f        Hopp,Daniel Andre Tande
cx:i0h2t3qafcsxgjil:2i6gc4c7zg5qg       bb8ff8365233ea91dfcdb36fdd84f87fcc33e1a8        utenriks,innenriks,trondheim,E6,midtbyen,bybrann,bilulykker
cx:imsxr7cak6xdzoy2:nybr893gy57d        325ee368538f17f9d61dfc24b2160f19e3e0d558        utenriks,innenriks,trondheim,E6,midtbyen,bybrann,bilulykker
cx:i1tqlab7r4mw8ei5:13kfx8w0qzqvr       935955483eb5ce55f3d181ff3e16e3d610d15207        Filmanmeldelser,Film,Mexico,Skrekkfilm
cx:iif4994p99hvuswo:1rs2rfz2fjous       f28b18132dfe141356c778b1909f13fcded3c613        Brattøra,Brattørkaia,Rockheim,Powerhouse
cx:ic3zj1lgz2zf6449:tzxn0dh5zzx4        f1b04d21c82146f30094d47b485ecc10ee277f0e        nyheter,trondheim,ukeadressa,eadressa,skattelister,dødsfall,tr
ønder,trafikk
cx:2swwnekfl2aup2t3olg5664iih:1k0q4cyr3dq56     8875d9afd4acbb01e43908b23170424a34cccce3        Sykehus,Rosenborg BK,Orkdal
cx:i4l2jq26hp2dlh8u:3q1ghpfa1xsv9       17a7ea777cb261130668d4314b1591f3767cd090        økonomi,finans,aksjer,skatt,børs,økonominyheter
cx:13117067288416798734470:30raldd1ek2xv 9da08a8be1d6cf85b0c7cd3f40b774cba66960b8       årets trønder,Birgit Skarstein,Haddy N'jie,tor olav mørseth,Me
rete Verstad,trøndelag,Levanger,Ivar Koteng,Stein Vanebo,Kristian Digre,trondheim concerts,Tor Singsaas
```

Figure 6. A part of the filtered dataset which includes user-id, item-id and keywords.

Evaluation Methods

For the performance evaluation of the recommender system you have developed, you should adopt the following 3 metrics:

1. **Precision, Recall and ROC**

   True positive (tp): the number of positive instances that are correctly predicted.
   True negative (tn): the number of negative instances that are correctly predicted.
   False negative (fn): the number of mispredicted negative instances.
   False positive (fp): the number of mispredicted positive instances.

- **Precision**
  Precision is used to measure the positive patterns that are correctly predicted from the total predicted patterns in a positive class. Precision can be defined as bellow.

$$precision=\frac{tp}{tp+fp}$$

- **Recall**
  Recall is used to measure the fraction of positive patterns that are correctly classified, which can be defined as:

$$recall=\frac{tp}{tp+tn}$$

- **F1-Score**
  This metric represents the harmonic mean between recall and precision values and can be denoted as bellow.

$$F1=\frac{2*precision*recall}{precision+recal}$$

Area under the Receiver Operating Characteristic (ROC) Curve (AUC)
AUC is one of the popular ranking type metrics. It was used to construct an optimized learning model and also for comparing learning algorithms. AUC value reflects the overall ranking performance of a classifier and can be calculated as bellow

$$AUC=\frac{S_p-n_p(n_n+1)/2}{n_p n_n}$$

where S_p is the sum of the all positive examples ranked, while n_p and n_n denote the number of positive and negative examples respectively.

2. **CTR**
   Click Through Rate (CTR) is the number of recommendations produced by a participating system that are clicked by users normalized by the total number of requests for recommendations that were sent to that system. Example: Participant "rocking recommendations" receives 100,000 recommendation requests. The system manages to provide valid, in-time suggestions in 95,000 cases. Users click on 4,500 suggestions. We compute a CTR of 4,500 / 100,000 = 4.5%.

So your recommender system should generate top N recommendations for a user and user click information (which user clicked to which article) can be found in the test dataset, so you can calculate the CTR.

For N we have set two different values: 10 and 20. Please calculate your CTR for both of these values and include in your final report.

3. **ARHR**
The third measure that is commonly used, is the average reciprocal hit rate (ARHR). This measure is designed for implicit feedback data sets, in which each value of $r_{uj} \in \{0,1\}$. Therefore, a value of $r_{uj}=1$ represents a “hit” where a customer has bought or clicked on an item. A value of $r_{uj}=0$ corresponds to a situation where a customer has not bought or clicked on an item. In this implicit feedback setting, missing values in the ratings matrix are assumed to be 0. Then, the ARHR metric for the user u is defined as follows:

$$ARHR(u) = \sum_{j \in I_u} \frac{r_{uj}}{v_j}$$

where v_j is the rank of item j in the recommended list, I_u represents the set of items rated by user u.

In your final report include a short paragraph about explaining the differences between these three evaluation methods you use.

Test dataset

We have prepared a test (and a training) dataset for you to evaluate your recommender system.

You can access the dataset here:
http://reclab.idi.ntnu.no/datasetfull/wi/test_one_week.gz

username: student
password: testtrain

Under this folder you can find the training dataset and a larger test dataset if you would like to try: http://reclab.idi.ntnu.no/datasetfull/wi

But for your final evaluations please use the test_one_week.gz

Tips for the group project

- Get familiar with the dataset. Learn about the available attributes and think about how you can use them in a recommender system. Some attributes may seem useful at first but when you look into the dataset in more detail, you may find out that the dataset does not contain enough number and variety of that attribute. So not all the attributes are useful.

- Choose the programming language you feel most comfortable with. Explore the existing libraries and tools available.

- Share the tasks with your group members and keep each other informed about your progress. Collaborative working environments like Github may be helpful.

- Pay attention to the final report. How you present your work is as important as your technical results.

- The recommender method is not limited to the one in the course textbook. You can use any state-of-the-art methods.

List of Some Useful Libraries and Tools for Recommender Systems
(in alphabetical order)

- Apache Mahout: https://mahout.apache.org/
- Apache Spark – Mlib: https://spark.apache.org/mllib/
- Deeplearn.js: https://deeplearnjs.org/
- GNU Octave and Octave-Forge for various scientific programming:
  https://www.gnu.org/software/octave/
  https://octave.sourceforge.io/packages.php
- GraphLab Create: https://turi.com/
- Keras – Deep learning library: https://keras.io/
- List of scikits (scikit-learn and surprise is also listed here): http://scikits.appspot.com/scikits
- Machine Learning in Python - scikit-learn: http://scikit-learn.org/stable/
- Natural Language Toolkit: http://www.nltk.org/
- Numpy – Scientific computing with Python: http://www.numpy.org/
- Pandas – Python Data Analysis Library: https://pandas.pydata.org
- Tensorflow: https://www.tensorflow.org/
- Theano: http://deeplearning.net/software/theano/
- Weka – Data mining in Java: https://www.cs.waikato.ac.nz/ml/weka/

More sources from the book, Recommender Systems: The Textbook, Charu C. Aggarwal:
http://charuaggarwal.net/Recommender-Systems.htm