

(A short guide to) Installing and Running Predictor and Searcher

Version 0.52

Rickard Cöster, Martin Svensson
{rick, martins}@sics.se

June 2, 2004

Contents

1	Introduction	1
1.1	Organization of this document	1
2	The files in the distribution	1
3	Using the command line interfaces	3
3.1	Searcher	3
3.2	Predictor	5
4	Predictor API and the core classes	8
4.1	Users	8
4.2	Document Objects	9
4.3	Predictors	9
5	The package se.sics.predictor and subpackages	9
5.1	Package overview	9
5.2	Core classes and interfaces	9
6	The package se.sics.ml and subpackages	10
6.1	Package overview	10
6.2	Core classes and interfaces	10
7	License, Bugs and Help	11
A	Example text document	12
B	Example user rating file	13
C	License	14

1 Introduction

This document describes how to install, run and use Predictor and Searcher.

Predictor is a system for collaborative and content-based filtering. The filtering functionality is fairly general, and not tied to a specific application domain; it may be used for classifying documents, recommending movies etc. Predictor can be used either as a stand-alone application or in a client-server environment. A stand-alone application will access the system through an API, whereas a client will access the system through Java servlets.

Searcher is a system for indexing and retrieval of text documents, using the Vector Space Model for document ranking. A set of shell scripts can be used to build an index, and a command driven program for searching the index.

1.1 Organization of this document

The rest of this document is organized as follows. An overview of the distribution and the files is given in Section 2. An overview of the command-line program for Predictor is given in Section 3.2, and the Searcher program in Section 3.1. The core classes of the Predictor classes are described in Section 4. The source package `se.sics.predictor` and its subpackages is described in Section 5 and the machine learning package `se.sics.ml` that is used by Predictor is described in Section 6. A short note regarding license, bugs and help is in Section 7. An example document that can be indexed with Searcher is listed in Appendix A. An example file containing user ratings for Predictor is listed in Appendix B. The license is listed in Appendix C.

2 The files in the distribution

First, get the files of the distribution from

- <http://www.sics.se/humle/socialcomputing/download>

The full distribution is contained in a single `.tar.gz` file that you unpack to a directory of your choice. There is no installation program, so the directory where you unpack the files is the installation directory. On the top level, the distribution contains two files and three folders:

- `Installation.pdf`
 - This file
- `license.txt`
 - A copy of the license file
- `bin/`
 - A directory that contains a set of scripts (`.bat`) for managing Predictor and Searcher
- `builds/`
 - A directory containing the source, compiled code, JavaDoc and other files for each of the six Java packages in the distribution

- `examples/`
 - A directory that contains example data for Predictor and Searcher

The six Java packages in the directory `builds/` are

- `collections/`
 - Java package for disc-based collections such as B+Trees.
- `disc/`
 - Java package including classes for disc-based storage and retrieval of objects (a form of serialization). Also included are algorithms for external sorting of data.
- `util/`
 - Java package for utility classes that do not fit into other packages.
- `ml/`
 - Java package for machine learning (classification and regression)
- `searcher/`
 - Java package for full text indexing and retrieval of documents. Includes classes for document sorting and building and searching an index.
- `predictor/`
 - Java package for collaborative and content-based filtering. Includes an API that has a number of methods for managing user profiles, documents, and machine learning programs to make predictions based on profiles or documents.

For each Java package directory, there are a number of files and subdirectories

<code>builds/<package>/src.jar</code>	JAR file containing the source files
<code>builds/<package>/<package>.jar</code>	JAR file containing the class files
<code>builds/<package>/classes/</code>	The java class files for the package
<code>builds/<package>/javadoc/</code>	JavaDoc documentation
<code>builds/<package>/lib/</code>	External libraries used by the package
<code>builds/<package>/src/</code>	The java source files for the package

The directory `examples/` contains example files that are helpful when using Predictor and Searcher

<code>examples/searcher/tiny.xml</code>	An XML file containing 3 documents
<code>examples/predictor/predictor1.xml</code>	An example startup file for Predictor
<code>examples/predictor/smallusers.xml</code>	A small set of users and ratings

3 Using the command line interfaces

Included in the distribution is a set of shell scripts and command-based programs that demonstrate the functionality of Predictor and the Searcher. In this section, we take a step-by-step tour of how these are used. The scripts included in the `bin/` directory need an environment variable called `BUILD_HOME` that must be set to the location of the directory `builds/` in your installation. In the examples that follow, this variable is set to `I:\projects\builds` since this is where the `builds/` directory is located on our machine.

3.1 Searcher

Before using Searcher to find words an index has to be present. A small example of a document set to index is given in the file `examples/searcher/tiny.xml` that contains (in a specific XML format) 3 short documents. The XML format is currently not finalized, so no DTD is given at this time. The tiny document set is also listed in Appendix A.

To index the tiny example, we need to extract the words and sort them. This is done by the script `bin/sortfile.bat`. Here is an example (where the distribution is located in `i:/projects`).

```
I:\projects\bin>sortfile.bat i:/projects/examples/searcher/tiny.xml sv i:/
```

The parameters are the input xml file, the Locale (`sv` for swedish) and temporary directory (`i:` in this case). After the file has been sorted, some new files have been created, most notably `i:/projects/examples/searcher/tiny.xml.key`

This file is then used when creating the index. To index write

```
I:\projects\bin>indexfile.bat i:/projects/examples/searcher/tiny.xml
```

The index have now been built. Start the search shell for this index by

```
I:\projects\bin>searchshell.bat i:/projects/examples/searcher/tiny.xml
```

The command-driven interface first prints some information and statistics regarding the index then waits for a command:

```
LOG started Mon Mar 22 19:03:51 CET 2004
Log file dir      = I:\projects\bin
num docs         = 3
avg unique terms = 28.33333333333332
pivot           = 28.33333333333332
slope           = 0.3
staticnorm      = 19.83333333333332
Commands:
(S) Search      word...
(G) Exp. search word...
(R) Range Search word...
(B) Range Boolean word...
```

(P) Expand Search word...
(L) Lookup word...
(A) Range Lookup word...
(K) Khi square word1 word2
(D) Doc docno...
(F) Freqinfo docno...
(T) Title docno...
(I) Info (no params)
(M) Write index file
(Y) CLEF Queries file
(Z) CLEF Stats file
(U) Write idf fileprefix
(W) Write docvecs fileprefix
(X) Settings (interactive)
(Q) Quit

cmd>

To search, type s and the search terms.

```
cmd>s alpha
hits.length = 2
0          3 0.043 test2
1          2 0.032 test1
```

The first column is the result index, the second is the document number, the third is the rank score and the last column is the document's title. In this case, document 2 with title "test2" was the best hit for this query. To view the contents of a document, type d and the document number

```
cmd>d 3
<docno   : -1>
<type    : type1>
<name    : test2>
<location: http://www.mydocuments.com/test2.xml>
<created : 2002-07-13-15:54:36-CEST>
<modified: 2002-07-13-16:25:20-CEST>
<meta    : the second best document>
<other   : >

<section name= AUTHOR weight = -1>
Alpha Beta Alpha Alpha Alpha
</section>
<section name= TITLE weight = -1>
Second document
</section>
<section name= TEXT weight = -1>
```

```
This is some text in the second document
</section>
<section name= CATEGORY weight = -1>
Test
</section>
```

To view the terms in a document, type f and the document number

```
cmd>f 3
1      2      4      ALPHA
1      2      1      BETA
1      2      2      DOCUMENT
1      2      1      IN
1      2      1      IS
1      1      2      SECOND
1      2      1      SOME
1      2      1      TEST
1      2      1      TEXT
1      2      1      THE
1      2      1      THIS
```

The first column can be ignored, the second column is the number of documents containing the term, the third is the total number of occurrences of the term.

To generate a set of document vectors from the index, type w and a file name

```
cmd>w i:/projects/examples/searcher/tiny.docvecs
```

Three files have now been created with the file name as prefix:

```
I:\projects\bin>dir ..\examples\searcher\*.docvecs*
```

```
Directory of I:\projects\examples\searcher
```

```
2004-03-22 19:11          1 996 tiny.docvecs.doclist.txt
2004-03-22 19:11          960 tiny.docvecs.docvecs.txt
2004-03-22 19:11          836 tiny.docvecs.wrdlist.txt
           3 File(s)          3 792 bytes
           0 Dir(s) 20 057 481 216 bytes free
```

The file docvecs.txt contains the document vectors in a format readable by the Predictor shell program. The doclist.txt wrdlist.txt contains the document contents and the mapping from number to terms used in the vectors.

To quit, type q. Note that all interaction is logged in a log file. A new log file is created during each session, so delete old log files on a regular basis if you don't need them.

3.2 Predictor

There is an example program using the Predictor API that is invoked by the script bin/predictorshell.bat. When opening a Predictor database, there is a setup XML file that contains the administrator's

name and password as well as path information for where to place certain files. An example XML file is located in `examples/predictor/predictor1.xml`. To open or create a new Predictor database, type the name of the XML file as first argument, the directory where the database reside or should be created as the second argument and the administrator's name and password.

```
I:\projects\bin>predictorshell.bat i:/projects/examples/predictor/predictor1.xml \  
i:/projects/examples/predictor admin adminpass
```

This will create a new set of files, display some statistics and start the command interface

```
created i:/projects/examples/predictor\predictor1\users\usermanager  
created i:/projects/examples/predictor\predictor1\documents\docmanager  
created i:/projects/examples/predictor\predictor1\objects\objectmanager  
Default feature predictor:  
id          = se.sics.ml.classifiers.CollaborativeFilter  
class      = se.sics.ml.classifiers.CollaborativeFilter  
params     = -k 50 -v 3  
istrained  = false  
bytesize  = 124
```

Commands:

```
uc (user create )      name pass  
ud (user delete )     name  
ui (user info  )     name  
ul (user list  )     namepattern  
uv (user vector )     name  
um (user modify )     name  
dc (doc create  )     name  
dv (doc view   )     name  
dm (doc modify  )     name  
dl (doc list   )     name  
ru (read users  )     file  
rd (read docs  )     file  
pf (pred.features)    name params  
h  (help       )  
q  (quit       )  
s  (save       )
```

cmd>

To create a user, type

```
cmd>uc user1 user1pass  
NamePass = user1-user1pass-null  
Created user user1      0      0.0
```

This will display the user as having an empty profile. To update the profile, type `um name`

```

cmd>um user1
Name      = user1-null
v = {  }
q to quit
idx val>1 3
idx val>2 4
idx val>3 3
idx val>q
v = { (1, 3) (2, 4) (3, 3)  }
v = user1      3      3.3333333

```

To read a set of users from a file, use

```

cmd>ru i:/projects/examples/predictor/smallusers.txt
Read 3 users

```

The contents of this user rating file is contained in Appendix B. To list allusers, type

```

cmd>ul
Users: 4
name    size    mean
U00001  5        4.0
U00002  5        2.4
U00003  4        3.25
user1   3        3.3333333

```

To make a Collaborative feature prediction (i.e. a recommendation), type

```

cmd>pf U00001
Name      = U00001-null
get vectors took 0 ms
user index is 0
prediction took 10 ms
Feature prediction:
[1,4.891] [2,3.527] [3,3.757] [4,3.317] [5,4.509]

```

To quit, type q. The database is saved and the log files are closed.

```

cmd>q
saving users.....
opened i:/projects/examples/predictor\predictor1\users\usermanager
.....done
saving docs.....
opened i:/projects/examples/predictor\predictor1\documents\docmanager
.....done
saving objects.....
opened i:/projects/examples/predictor\predictor1\objects\objectmanager
.....done
saving features.....done

```

```
saving request log.....done
saving error log.....done
```

To open the database again, use the same command as when creating it in the first place.

4 Predictor API and the core classes

The Predictor API is accessed via a Java class, and can be used in any stand-alone application supporting Java. The JavaDoc documentation located in `builds/predictor/javadoc/index.html` should be used to locate the documentation for the class `se.sics.predictor.PredictorApi` and other classes. Also, look into the source code for the Predictor command interface program. It is located in `se.sics.predictor.admin.AdminShell`.

Users are fundamental objects in Predictor, since the purpose of the system is to filter and find relevant information for users. Each user has a name, a password and a profile that describe that user's interest, relevance judgments or preferences regarding some information. The profile often contains the user's explicit or implicit actions, such as votes for movies or books in a classical collaborative filtering system. But the profile may also contain content information such as keyword or document weights, for document filtering. The API contains a number of functions for maintaining user profiles.

Since the information subject for filtering vary between applications, it is impossible to store an explicit model of this information in Predictor. Instead, Predictor store information handles, and it is the responsibility of the application program to keep track of the connection between handles and information.

Information filtering is performed by invoking a machine learning program that takes a user and its profile as arguments. The task of such a program is to predict, for a user, the values of a set of information handles. Internally, these programs are called Predictors. A Predictor may be a classifier, clusterer or any other algorithm that can take users and their profiles as input and predict a value for one or more information handles.

4.1 Users

All user objects contain the user's name, password and profile. User name and password is required when accessing the API.

The user profile is a vector of features, where a feature is a tuple of an ordinal number and a real valued weight. The ordinal number is a mapping to some physical object, determined by the application, and the weight is a record of the user's interest, action or usage pattern on that object. In the case of a movie recommender system, a feature may contain the ordinal number of a movie and the weight would be the user's vote for that movie, a keyword or phrase.

The API provides several functions for user management. Users can be created, deleted, updated and inspected.

The feature vector representation of user profiles limits the expressiveness of user interest. A more sophisticated model of user interest is obtained by the use of Predictors.

4.2 Document Objects

Document objects contain a name and a document vector. The document vector is implemented using the same feature vector classes that are available for the profile vectors.

The API contains methods for viewing, creating and deleting Document objects. The methods that change or update Document objects are only accessible to the administrator.

4.3 Predictors

Predictors are machine-learning programs that take a feature vector as input and produces predictions for a set of information handles. Users can invoke but not update, remove or in any other way alter a Predictor's internal model. Predictors are added by the administrator.

Two predictors are implemented and shipped with the package. The first is a memory-based collaborative filtering algorithm, using standard methods such as Pearson correlation, inverse user frequency and default voting. The second is a Support Vector Machine classifier. In a real scenario, each user should be assigned a document classifier that is trained to classify documents according to the user's profile.

The API contains methods for adding and deleting Predictors. Since machine learning is an inherently iterative process, it is convenient to train such a program off-line and, when satisfied with its performance, add it to the system.

5 The package `se.sics.predictor` and subpackages

This package contains an API that supports personalized information filtering. The API may be access directly via a Java class (`PredictorApi`) or via client (`PredictorClient`) that access the API via a servlet server.

User profiles are represented by feature vectors, and predictors that operate on user profiles may be added via the API by the administrator.

5.1 Package overview

The predictor package has six sub packages. At the top level, there are classes for user and object management and the API. The package `predictor.cmd` contains commands such as create, delete and update users and generating predictions. `predictor.servlet` contains the corresponding servlet classes for the commands. The client API and the XML parsing methods are placed in the package `predictor.client`.

5.2 Core classes and interfaces

`se.sics.predictor.PredictorAPI` `PredictorApi` supports personalized information filtering. It provides methods for user management and inspection, as well as making predictions on information handles on the basis of user profiles.

`se.sics.predictor.PredictorClient` `PredictorClient` access the `PredictorAPI` via a servlet server, and handles all communication as well as the parsing of the XML response from the server.

se.sics.predictor.User User is the object that represents a user of the PredictorAPI. Contains name, password and a profile.

se.sics.predictor.Profile A Profile is a feature vector representation of a user's interest, relevance judgements or preferences regarding some information.

se.sics.predictor.UserManager Manages User objects in a disk-based database.

se.sics.predictor.ObjectManager Manages Predictor objects in a disk-based database structure.

6 The package `se.sics.ml` and subpackages

This package contains classes for performing machine learning on data that can be represented as mathematical vectors.

6.1 Package overview

The `ml` package has a few top-level classes, and three subpackages. At the top level, there are abstract classes for classification and clustering. The subpackage `ml.core` contains classes for vectors, matrices and nearest neighbor searching. The packages `ml.classifiers` contains implementations of classifier algorithms, and the subpackage `ml.eval` contains classes for evaluating the performance of classifiers.

6.2 Core classes and interfaces

se.sics.ml.core.MLVector A `MLVector` is a vector of `MLVector.Feature` objects containing an integer index and a double value. This is the representation of a mathematical vector and is used for representing examples in the machine learning methods. The fundamental methods are to put, get or remove an index-value pair from the vector. There are several other methods such as iterating, sorting and converting a vector. `MLVector` is an abstract class, and defines the functionality but not the implementation. In the package, there are subclasses with different implementations: as dense arrays, sparse arrays or as hash maps.

se.sics.ml.core.MLMatrix `MLMatrix` is an array of `MLVector` objects, arranged as row vectors. It is not an implementation of mathematical matrix but rather a convenient container of an ordered list of `MLVector` objects. It contains several utility methods such as calculating statistics from the matrix.

se.sics.ml.core.VectorSim `VectorSim` is an interface for determining the similarity value between two vectors, and also how the similarity values should be ordered. Included in the package are classes that implement different measures such as Euclidean distance, the Cosine of the angle, Pearson correlation and several more.

se.sics.ml.core.NearestNeighbor NearestNeighbor is a class for finding, among a set of MLVector objects, the k nearest neighbours to a MLVector by the use of a specific VectorSim object.

se.sics.ml.Predictor Predictor is an abstract base class of all classifiers and clusterers, and defines methods for passing parameters to the machine-learning program.

se.sics.ml.VectorClassifier VectorClassifier defines methods for supervised learning of vectors. It includes methods for training the classifier as well as predicting the value or class of a test example.

se.sics.ml.classifiers.SVMClassifier/BPNCClassifier SVMClassifier is a VectorClassifier implementation of a Support Vector Machine. It uses the third-party library [LIBSVM](#) and implements both support vector classification and regression. BPNCClassifier is an implementation of a feed-forward multilayer neural network, using backpropagation and momentum for weight updating.

se.sics.ml.FeatureClassifier A FeatureClassifier is a classifier for features, as opposed to vectors. The goal of feature classification is to predict the values of certain vector features, given the features of other vectors.

se.sics.ml.classifiers.CollaborativeFilter CollaborativeFilter is a descendant of FeatureClassifier, which predicts feature values for a vector on the basis of previously stored vectors and their feature values. It implements memory-based collaborative filtering, where user votes and opinions are used to group similar users together for making predictions.

7 License, Bugs and Help

This software is distributed because we believe it can be of interest to other researchers and the industry. We do not claim that it is correct or even suited for any purpose, but you are free to use it as long as you follow the license text. The license can be found in the file `license.txt` and in the beginning of every source code file. The software uses a third party library [LIBSVM](#), guarded by another license. This license file is located in the same directory as the LIBSVM library in this distribution.

This software has bugs. When you find a bug, you can help us by reporting it. It is however, not our responsibility to correct it even though we will try to do so. To report a bug, we will set up a bug reporting facility on the software's web site.

We will not in general answer questions regarding the software, neither via email or other forums, since we simply do not have the time. If we find that there is a considerate interest in this software we will seriously consider setting up a more elaborate web site with discussion forums, bug tracking system and other features that can be of help for developers.

A Example text document

```
<?xml version='1.0' encoding='iso-8859-1' standalone="yes"?>
<documents numdocs="3">
<document type="newsprint" name="fartygsolycka"
  location="1994/01/02/19940102140188fartygsolycka.ttt.xml"
  created="2003-03-31-17:40:30-CEST" modified="2003-03-31-17:40:30-CEST"
  meta="" other="">
<section name="TITLE">
  lastfartyg last fartyg 36 sjunka nordatlant nord atlant </section>
<section name="INGRESS">
  japansk lastfartyg last fartyg 36 besättning sjunka lördag
  nordatlant nord atlant drygt 100 mil newfoundland
</section>
<section name="TEXT">
  275 meter lång bulkfartyg bulk fartyg marika 7 lastad järnmalm järn
  malm väg kanada holland hamna hård väder sent lördag sända besättning
  nödsignal nöd signal talesman sjöräddningscentral sjö räddning central
  halifax uppge fartyg troligtvis sjunka råka meter hög våga
  herculesplan kanadensisk flygvapen flyg vapen påbörja spaning fartyg
  område dirigera riktning marika 7 : s sen angiven position
  flygplansbesättning flyg plan besättning meddela upptäckt ljussignal
</section> </document>
<document type="type1" name="test1"
  location="http://www.mydocuments.com/test1.xml"
  created="2001-07-13-14:54:36-CET" modified="2001-07-13-15:25:20-CET"
  meta="the best document" other="">
  <section name="AUTHOR">Alpha Beta</section>
  <section name="TITLE">First document</section>
  <section name="TEXT">This is some text in the first document</section>
  <section name="CATEGORY">Test</section>
</document>
<document type="type1" name="test2"
  location="http://www.mydocuments.com/test2.xml"
  created="2002-07-13-14:54:36-CET"
  modified="2002-07-13-15:25:20-CET"
  meta="the second best document" other="">
  <section name="AUTHOR">Alpha Beta Alpha Alpha Alpha</section>
  <section name="TITLE">Second document</section>
  <section name="TEXT">This is some text in the second document</section>
  <section name="CATEGORY">Test</section>
</document>
</documents>
```

B Example user rating file

1	1	5	978300799
1	2	3	978302199
1	3	3	978301999
1	4	4	978300299
1	5	5	978824299
2	1	1	978300799
2	2	3	978302199
2	3	3	978301999
2	4	4	978300299
2	5	1	978824299
3	1	4	978300799
3	2	3	978302199
3	3	3	978301999
3	4	3	978300299

C License

```
/*
 * Copyright (c) 2000-2004, Rickard Cöster, Martin Svensson
 * All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are met:
 *
 * Redistributions of source code must retain the above copyright notice,
 * this list of conditions and the following disclaimer.
 * Redistributions in binary form must reproduce the above copyright notice,
 * this list of conditions and the following disclaimer in the documentation
 * and/or other materials provided with the distribution.
 * Neither the name of SICS nor the names of its contributors
 * may be used to endorse or promote products derived from this software
 * without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
 * LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
 * CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
 * SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
 * INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
 * CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
 * ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 */
```