Privacy in Multiple On-line Social Networks – Re-identification and Predictability

This software demonstrates how overlapping Online Social Networks can affect user privacy. The full paper will be published soon and will be made available:

> *David F. Nettleton, Vladimir Estivill-Castro, Julian Salas, "Privacy in Multiple On-line Social Networks – Re-identification and Predictability", Transactions on Data Privacy (in press).*

The main github repository, "**OverlappingOSNsUserPrivacy**", contains three Java projects:

- **IdentifyCrossSAN**
- **SplitSAN**
- **SANPreprocData**

The first two correspond to the empirical Section 5.2 (Re-identification using a Set-theoretical Approach) and the third to Section 5.3 (Predictive Modeling Approach using SAN Metrics) of the paper, respectively.

The project "SANPreprocData" is copyright David F. Nettleton (GNU General Public License V3.0) and the projects "IdentifyCrossSAN", "SplitSAN" are copyright Vladimir Estivill-Castro (GNU General Public License V3.0).

SANPreprocData

All the following explanation refers to the project SANPreprocData.

This program essentially preprocesses the raw graph data (structure in one file and data one record per user) for its use in Weka to predict based on different dependent/independent attribute combinations. First it inputs the graph into an appropriate data structure, then it calculates the SAN metrics, and then creates the flags which indicate if a user has or not a given attribute. This file is then output to disc and is used as input to Weka in which we used the J48 rule induction algorithm and SVM-SMO algorithm to build supervised models. Models were built for different combinations of metrics/predictors.

The main input files are "SEP4.csv" and "ATTRIBDATA.csv".

An intermediary output/input file is generated, called "flag_ALL.csv" which is then used in the remainder of the process.

The output files are:

"data_all_weka_G_topidN.csv"
"data_all_weka_G_topattN.csv"

Where G is the graph type, ALL, SG1 or SG2 and N is the number 0 to 4.

This enables different combinations which are then input to Weka in order to predict the given topid (user) or topatt (attribute).

The choice of which are the top attributes and users is controlled in the code in the routine "readData/createFlagsV3Filter.java". The topids are found statistically (not in Java program) and then at lines 130 and 156 you will see the id assignments hard coded.

SAN metrics:
These are calculated by methods associated with the user class in User/User.java, see starting line 287 for the non-SAN metrics (cn, cnd, aa, aad) and starting at line 373 for the SAN metrics (cnsan, cnsand, aasan, aasand). The "d" versions (e.g. cn and cnd) are the discretized versions, that is, they use a category output.

Whole graph, subgraphs
The choice of which graph to write (ALL, SG1, SG2), that is, whole graph, sub-graph 1 or sub-graph2 is controlled in the code of the main program "SANPreprocData.java" by calling the routines:

"createAttributes/createAttributesALL.java",
"createAttributes/createAttributesSG1.java",
"createAttributes/createAttributesSG2.java",

respectively.

Java project structure for SANPreprocData
The dependencies are as follows:

The main program, SANPreprocData.java , calls the other programs/routines in the following order:

readData/readData.java	readTopology()
readData/loadUsers.java	(calls user class methods)
readData/readData.java	processAttributes()

readData/createFlagsV3filter.java

readData/readData.java	writeInstancesforWeka()
readData/readData.java	writeDataUsers()
readData/readData.java	writeDataAttributes()
readData/readData.java	writeUserGraph()
readData/readData.java	writeAttributeGraph()

createAttributes/createAttributesGALL.java	(see lines starting 88, 172 which call metric calculation methods)
createAttributes/createAttributesSG1.java	(see lines starting 88, 172 which call metric calculation methods)
createAttributes/createAttributesSG2.java	(see lines starting 88, 172 which call metric calculation methods)