# Difference between Communities Opinions Algorithm – Diffcomm

## Repository Info

The source code is available at: https://diffcomm.googlecode.com/svn

## Development Environment

1. **Install Eclipse IDE** and proper plugins that can be found at: http://www.eclipse.org/downloads/

2. **Install Subversive plugin and SVN Connectors:**
   a. Follow: http://www.polarion.com/products/svn/subversive.php?src=eclipseproject
   b. Install the latest:
      i. Add update site http://download.eclipse.org/technology/subversive/0.7/update-site/ and install the Team providers and the Integration plugins.
      ii. Add update site http://community.polarion.com/projects/subversive/download/eclipse/2.0/update-site/ and install the SVN connectors, the native JavaHLand the SVNKit implementations.
   c. In the preferences of SVN for eclipse, select the SVN Kit as the connector (Window - Preferences - Team - SVN - SVN Connector)

3. **Install Apache-ant-1.8.1:** http://ant.apache.org/

## Configuring the Project

Create a **New Project** in Eclipse:

a. Select SVN – Project from SVN – selecting the SVN repository given above.
b. Choose resource: *diffcomm*.
c. Checkout as a Project in Workspace.

## Data Files

An example of data files can be found in the *data* directory of the project.

1. **Create a folder** that will contains our data files
2. Enter in the folder and **create the files**:
   ***a. Candidates.csv:*** *contains a list of candidates*
      i. The <u>first line</u> of the file must be:
      **"CANDIDATE","NAME"** and type RETURN;

      ii. The <u>others lines</u>:
      **candidate id, "candidate name"** and type RETURN

      where
      - candidate id: ascending numbers ≥ 1, all different;
      - "candidate name": ex. "Mario Rossi"

*Example*:

```
"CANDIDATE","NAME"
1,"Mario Rossi"
2,"Roberto Bianchi"
```

***b.*** ***Communities.csv:*** *contains a list of communities*

i. The first line of the file must be:
**"COMMUNITIES","NAME"** and type RETURN;

ii. The others lines:
**"community id", "community name"** and type RETURN

where
- "community id": ex. "a", "b", …
- "community name": ex. "[edoc]", "pippo", …

*Example*:

```
"COMMUNITIES","NAME"
"a","[edoc]"
"b","[sat]"
"c","pippo"
```

***c.*** ***Voter.csv:*** *contains a list of voters*

i. The first line of the file must be:
**"VOTER","NAME"** and type RETURN;

ii. The others lines:
**voter id, "voter name"** and type RETURN

where
- voter id: ascending numbers ≥ 1, all different;
- "voter name": ex. "Martina Verdi", …

*Example*:

```
"VOTER","NAME"
1,"Martina Verdi"
2,"Voter2"
3,"Voter3"
```

***d.*** ***VoterMembership.csv:*** *contains the list of voter-community relation*

i. The first line of the file must be:
**"VOTER","COMMUNITY","NUMBERS_OF_PUBBL"** and type RETURN;

ii. The others lines:
**voter id, "community id",numbers of pubblications** and type RETURN

where

- voter id: id of a voter;
- “community id”: id of the community of the voter;
- number of pubblications: numbers ≥ 1, 1 if not specified. This rapresents the number of pubblication of a voter in a community;

*Example*:

```
“VOTER”,”COMMUNITY”,”NUMBERS_OF_PUBBL”
1,”a”,2
2,”a”,2
2,”b”,4
3,”c”,1
```

**!!! IMPORTANT !!!** A voter may belong to more communities. In this case, add a line for voter-community relation as showed in the example (see above).

***e.*** ***Opinions.csv:*** *contains the list of opinions*

i. The first line of the file must be:
**“VOTER”,”CANDIDATE”,”VOTE”** and type RETURN;

ii. The others lines:
**voter id,candidate id,vote** and type RETURN

where
- voter id: id of a voter;
- candidate id: id of a candidate;
- vote: numbers ≥ 0, 0 if not specified. This rapresents the vote of a voter on a candidate;

*Example*:

```
“VOTER”,”CANDIDATE”,”VOTE”
1,1,2
1,2,4
2,1,0
2,2,1
3,1,2
3,2,3
```

**!!! IMPORTANT !!!** Insert an opinion **for each pair voter-candidate** that can be found. If the value an opinions is not specified intert 0 as vote.

For each of this files is **IMPORTANT**:

- Type RETURN **after all lines except the last** line;
- In each row **do not put spaces** before and/or after the commas.

## Compile with Ant

1. Open a shell and **go to the project folder**.
2. Type:
   - **`ant compile`** to compile the project;
     This command create the *target* folder and creates all .class files.

   - **`ant dist`** to generate the jar executable file.
     This command create the *dist* folder and the *dist/Diffcomm.jar* executable file.

Another ant command is available(**`ant clean`**). If you type **`ant`** in the project folder, a list of available target is showed.

## Run Diffcomm.jar

1. Open a shell and go to the folder where you have the **Diffcomm.jar file**.
2. Type:

   `java -jar Diffcomm.jar` ***`arg1`***

   where ***arg1*** is the path of data folder.
3. The **Result.csv** file is in data folder.

Example:

```
pippo@pippo: ~ $ ls
Documents       Diffcomm.jar

pippo@pippo: ~ $ cd Documents/diffcomm_data
pippo@pippo: ~/Documents/diffcomm_data $ ls
Candidates.csv Opinions.csv        Voter.csv
Communities.csv     VoterMembership.csv

pippo@pippo: ~/Documents/diffcomm_data $ cd
pippo@pippo: ~ $ java -jar Diffcomm.jar
/home/pippo/Documents/diffcomm_data/
log4j:WARN No appenders could be found for logger
(dk.eobjects.metamodel.CsvDataContextStrategy).
log4j:WARN Please initialize the log4j system properly.

The document Result.csv has been created at
/home/pippo/Documents/diffcomm_data

pippo@pippo: ~ $ pippo@pippo: ~ $ cd
Documents/diffcomm_data
pippo@pippo: ~/Documents/diffcomm_data $ ls
Candidates.csv Opinions.csv            Voter.csv
Communities.csv     VoterMembership.csv   Result.csv
```