PEGASUS User's Guide

PROJECT PEGASUS

Contents

1. Overview

1) General Information

PEGASUS: Peta-Scale Graph Mining System

Version: 2.0

Date: Sep. 6th, 2010

Authors: U Kang, Duen Horng Chau, and Christos Faloutsos

Carnegie Mellon University

PEGASUS is a Peta-scale graph mining system on hadoop, fully written in Java.
It computes the degree distribution, PageRank, RWR(Random Walk with Restart) scores, radii/diameter, and connected components of very large graphs with more than billions of nodes and edges.
The details of PEGASUS can be found in the following paper:

U Kang, Charalampos E. Tsourakakis, and Christos Faloutsos.
PEGASUS: A Peta-Scale Graph Mining System - Implementation and Observations.
IEEE International Conference On Data Mining 2009, Miami, Florida, USA.

If your work uses or refers to PEGASUS, please cite the papers using the following bibtex entries:

```
@article{PegasusICDM2009,
title = {PEGASUS: A Peta-Scale Graph Mining System - Implementation and Observations},
author = { Kang, U and Tsourakakis, C.E and Faloutsos, C.},
year = {2009},
journal = {IEEE International Conference On Data Mining},
}

@article{PegasusKAIS,
title = {PEGASUS: Mining Peta-Scale Graphs},
author = { Kang, U and Tsourakakis,C.E and Faloutsos,C.},
year = {2010},
journal = {Knowledge and Information Systems},
}
```

For more information and demo, visit PEGASUS homepage http://www.cs.cmu.edu/~pegasus
For questions on PEGASUS, contact <ukang@cs.cmu.edu>.

2) License

If you use PEGASUS for research or commercial purposes, please let us know
your institution(company) and whether it's ok to mention it among the users of PEGASUS.

2. Installation

This section guides you the installation process of PEGASUS.

1) Environment

PEGASUS can be run in any machine that supports hadoop, but the shell scripts and
code packaging scripts works easily in Linux or Unix machines.
PEGASUS needs the following softwares to be installed in the system:

- Hadoop 0.20.1 or greater from http://hadoop.apache.org/
- Apache Ant 1.7.0 or greater from http://ant.apache.org/
- Java 1.6.x or greater, preferably from Sun
- Python 2.4.x or greater
- Gnuplot 4.2.x or greater

2) Download

Download the installation file 'PEGASUSH-2.0.tar.gz' from http://www.cs.cmu.edu/~pegasus
Extract the file, then the directory 'PEGASUS' will be created.
Cd to the PEGASUS directory, then you are done.

3) Configurations

PEGASUS requires Hadoop to be installed in the system.
The typical usage of PEGASUS is to run it on a gateway server of the Hadoop cluster.
Alternatively, you can install Hadoop on a single machine and run PEGASUS on the single machine.
In any case, the `hadoop` command should be runnable in the PEGASUS installation directory.
For more information on setting up Hadoop, refer to the following web pages:

- single machine: http://hadoop.apache.org/common/docs/r0.21.0/single_node_setup.html
- cluster: http://hadoop.apache.org/common/docs/r0.21.0/cluster_setup.html

PEGASUS requires additional four programs to run. They are Java, Apache Ant, Python, and Gnuplot.

In essence, the binaries 'java', 'ant', 'python', and 'gnuplot' should be accessible from the PEGASUS installation directory. No additional configurations are needed for these programs. The purposes of these programs are:

- Java is required since Hadoop runs on top of Java.
- Apache Ant is needed to rebuild the code after you modify it.
- Python and Gnuplot is needed to generate plots in the interactive command line UI pegasus.sh.

3. Running

This section guides you how to run PEGASUS, from preparing input to running commands.

1) Preparing Graph

PEGASUS works on graphs with TAB-separated plain text format. Each line contains the source and destination node id of an edge. The node id starts from 0. For example, here is an example graph ‘catepillar_star.edge’ which is included in the installation file. It has 16 nodes.

```
0	1
1	2
1	3
3	4
3	6
5	6
6	7
6	8
6	9
10	11
10	12
10	13
10	14
10	15
```

2) PEGASUS Interactive Shell

There are two ways of running PEGASUS. The first is to run it with an interactive command line UI `pegasus.sh`, and the other is to run individual algorithms by corresponding commands. We recommend to use the interactive UI for beginners since it is convenient to use and provides plotting functions. On the other hand, advanced users might want to use individual commands for custom settings.

To start the PEGASUS shell, type `pegasus.sh` in the command line. Here is the list of the possible commands in the shell.

Command	Description
add [file or directory] [graph_name]	upload a local graph file or directory to HDFS
del [graph_name]	delete a graph
list	list graphs
compute ['deg' or 'pagerank' or 'rwr' or 'radius' or 'cc'] [graph_name]	run an algorithm on a graph
plot ['deg' or 'pagerank' or 'rwr' or 'radius' or 'cc' or 'corr'] [graph_name]	generate plots
help	show this screen
demo	show demo
exit	exit PEGASUS

If you use the `compute` command to run algorithms, the result is saved under the HDFS directory pegasus/graphs/[GRAPH_NAME]/results/[ALGORITHM_NAME].

Type `demo` in the shell to see the demo. The demo adds a graph file 'catepillar_star.edge' by the name 'catstar' in PEGASUS, computes the inout degree, and generates the degree distribution plot 'catstar_deg_inout.eps'.

For more detailed demos of using the PEGASUS shell, visit the PEGASUS homepage at http://www.cs.cmu.edu/~pegasus

3) Running individual scripts

You can also run each algorithm with corresponding shell scripts. Here is the list of the algorithm, shell scripts, and corresponding demo scripts.

Algorithm	Running Script	Demo Script
Degree	run_dd.sh	do_dd_catstar.sh
PageRank-plain	run_pr.sh	do_pr_catstar.sh
PageRank-block	run_prblk.sh	do_prblk_catstar.sh
RWR-plain	run_rwr.sh	do_rwr_catstar.sh
RWR-block	run_rwrblk.sh	do_rwrblk_catstar.sh
Radius-plain	run_hadi.sh	do_hadi_catstar.sh
Radius-block	run_hadiblk.sh	do_hadiblk_catstar.sh

Connected Component-plain	run_ccmpt.sh	do_ccmpt_catstar.sh
Connected Component-block	run_ccmptblk.sh	do_ccmptblk_catstar.sh

The difference of –plain and –block algorithm is that the –block algorithm uses block-encoding which results in faster running time.

The ‘running script’ is the script you need to use to run each algorithm. It requires several parameters which will be described next. The ‘demo script’ is the script to tell you how to use the ‘running script’. The demo scripts do not require any parameters. Just type the demo script name and it will run. Or, if you simply type `make` in the installation directory, the demo of running radius algorithm on the example graph ‘catepillar_star.edge’ will be executed.

3.1) Degree Distribution

To run Degree Distribution, you need to do the two things:
- copy the graph edge file to a HDFS directory, say dd_edge
- execute ./run_dd.sh

The edge file is a plain text file where each line is in SRC_ID TAB DST_ID format.
The range of node id is from 0 to number_of_nodes_in_graph - 1.

The syntax of run_dd.sh is:

./run_dd.sh [in or out or inout] [#_of_reducers] [HDFS edge_file_path]
[in or out or inout] : type of degree to compute.
[#_of_reducers] : number of reducers to use in hadoop.
- The number of reducers to use depends on the setting of the hadoop cluster.
- The rule of thumb is to use (number_of_machine * 2) as the number of reducers.

[HDFS edge_file_path] : HDFS directory where edge file is located

ex: ./run_dd.sh inout 16 dd_edge

The outputs of Degree Distribution are saved in the following HDFS directories:
dd_node_deg :
- Each line contains the degree of each node in the format of
  (nodeid TAB degree_of_the_node).

- For example, the line "1	3" means that the degree of node 1 is 3.

dd_deg_count : The degree distribution.
- Each line contains a degree and the number of nodes with the degree.
- For example, the line "1	12" means that 12 nodes have degree 1.

The working example of running Degree Distribution is in the do_dd_catstar.sh.

3.2) PageRank-plain

To run PageRank-plain, you need to do the two things:
- copy the graph edge file to the HDFS directory pr_edge
- execute ./run_pr.sh

The edge file is a plain text file where each line is in SRC_ID TAB DST_ID format.
The range of node id is from 0 to number_of_nodes_in_graph - 1.

The syntax of run_pr.sh is:

./run_pr.sh [#_of_nodes] [#_of_reducers] [HDFS edge_file_path] [makesym or nosym]
[#_of_nodes] : number of nodes in the graph
[#_of_reducers] : number of reducers to use in hadoop.
- The number of reducers to use depends on the setting of the hadoop cluster.
- The rule of thumb is to use (number_of_machine * 2) as the number of reducers.

[HDFS edge_file_path] : HDFS directory where edge file is located
[makesym or nosym] : makesym-duplicate reverse edges, nosym-use original edge file
- When the input graph is directed and you want to calculate directed radii/diameter, then use 'nosym' in the 4th parameter.
- When the input graph is directed and you want to calculate undirected radii/diameter, then use 'makesym' in the 4th parameter.
- When the input graph is undirected, use 'nosym' in the 4th parameter.

ex: ./run_pr.sh 16 3 pr_edge makesym

The output of PageRank-plain is saved in the following HDFS directory:
pr_vector :
- Each line contains the PageRank of each node in the format of

(nodeid TAB "v"PageRank_of_the_node).

- For example, the line "1 v0.10231778333763829" means that the PageRank of node 1 is 0.10231778333763829.

pr_minmax : The minimum and the maximum PageRank.
- The minimum PageRank is the second column of the line that starts with "0".
- The maximum PageRank is the second column of the line that starts with "1".

pr_distr : The histogram of PageRank. It divides the range of (min_PageRank, max_PageRank) into 1000 bins and shows the number of nodes which have PageRanks that belong to such bins.

The working example of running PageRank-plain is in the do_pr_catstar.sh.
If you run do_pr_catstar.sh, it will run for 32 iterations until it converges.

3.3) PageRank-block

To run PageRank-block, you need to do the two things:
- copy the graph edge file to the HDFS directory pr_edge
- execute ./run_prblk.sh

The edge file is a block text file where each line is in SRC_ID TAB DST_ID format.
The range of node id is from 0 to number_of_nodes_in_graph - 1.

The syntax of run_prblk.sh is:

./run_prblk.sh [#_of_nodes] [#_of_reducers] [HDFS edge path] [makesym or nosym] [block width]

[#_of_nodes] : number of nodes in the graph

[#_of_reducers] : number of reducers to use in hadoop.
- The number of reducers to use depends on the setting of the hadoop cluster.
- The rule of thumb is to use (number_of_machine * 2) as the number of reducers.

[HDFS edge_file_path] : HDFS directory where edge file is located

[makesym or nosym] : makesym-duplicate reverse edges, nosym-use original edge file
- When the input graph is directed and you want to calculate directed PageRank, then use 'nosym' in the 4th parameter.
- When the input graph is directed and you want to calculate undirected PageRank, then use 'makesym' in the 4th parameter.
- When the input graph is undirected, use 'nosym' in the 4th parameter.

[block_width] : block width. usually set to 16.

ex: ./run_prblk.sh 16 3 pr_edge makesym 2

The output of PageRank-block is saved in the following HDFS directory:
pr_vector :
- Each line contains the PageRank of each node in the format of
  (nodeid TAB "v"PageRank_of_the_node).
- For example, the line "1 v0.10231778333763829" means that the PageRank of node 1 is 0.10231778333763829.

pr_minmax : The minimum and the maximum PageRank.
- The minimum PageRank is the second column of the line that starts with "0".
- The maximum PageRank is the second column of the line that starts with "1".

pr_distr : The histogram of PageRank. It divides the range of (min_PageRank, max_PageRank) into 1000 bins and shows the number of nodes which have PageRanks that belong to such bins.

The working example of running PageRank-block is in the do_prblk_catstar.sh.
If you run do_prblk_catstar.sh, it will run for 32 iterations until it converges.

3.4) RWR-plain

To run RWR-plain, you need to do the two things:
- copy the graph edge file to the HDFS directory rwr_edge
- execute ./run_rwr.sh

The edge file is a plain text file where each line is in SRC_ID TAB DST_ID format.
The range of node id is from 0 to number_of_nodes_in_graph - 1.

The syntax of run_rwr.sh is:

./run_rwr.sh [HDFS edge_file_path] [query path] [#_of_nodes] [#_of_reducers] [makesym or nosym] [new or contNN] [c]
[HDFS edge_file_path] : HDFS directory where edge file is located
[query path] : HDFS directory containing query nodes
[#_of_nodes] : number of nodes in the graph

[#_of_reducers] : number of reducers to use in hadoop.
- The number of reducers to use depends on the setting of the hadoop cluster.
- The rule of thumb is to use (number_of_machine * 2) as the number of reducers.

[makesym or nosym] : makesym-duplicate reverse edges, nosym-use original edge file
- When the input graph is directed and you want to calculate directed RWR scores, then use 'nosym' in the 4th parameter.
- When the input graph is directed and you want to calculate undirected RWR scores, then use 'makesym' in the 4th parameter.
- When the input graph is undirected, use 'nosym' in the 4th parameter.

[new or contNN] : starts from scratch, or continue from the iteration NN
[c] : mixing component. Default value is 0.85.

ex: ./run_rwr.sh rwr_edge rwr_query 16 3 nosym new 0.85

The output of RWR-plain is saved in the following HDFS directory:
rwr_vector :
- Each line contains the RWR of each node in the format of
  (nodeid TAB "v"RWR_of_the_node).
- For example, the line "1 v0.10231778333763829" means that the RWR of node 1 is 0.10231778333763829.

rwr_minmax : The minimum and the maximum RWR.
- The minimum RWR is the second column of the line that starts with "0".
- The maximum RWR is the second column of the line that starts with "1".

rwr_distr : The histogram of RWR. It divides the range of (min_RWR, max_RWR) into 1000 bins and shows the number of nodes which have RWRs that belong to such bins.

The working example of running RWR-plain is in the do_rwr_catstar.sh.
If you run do_rwr_catstar.sh, it will run for 19 iterations until it converges.

3.5) RWR-block

To run RWR-block, you need to do the two things:
- copy the graph edge file to the HDFS directory rwr_edge

- execute ./run_rwrblk.sh

The edge file is a block text file where each line is in SRC_ID TAB DST_ID format.
The range of node id is from 0 to number_of_nodes_in_graph - 1.

The syntax of run_rwrblk.sh is:

./run_rwrblk.sh [HDFS edge_file_path] [query path] [#_of_nodes] [#_of_reducers] [makesym or nosym] [block width] [c]
[HDFS edge_file_path] : HDFS directory where edge file is located
[query path] : HDFS directory containing query nodes
[#_of_nodes] : number of nodes in the graph
[#_of_reducers] : number of reducers to use in hadoop.
 - The number of reducers to use depends on the setting of the hadoop cluster.
 - The rule of thumb is to use (number_of_machine * 2) as the number of reducers.

[makesym or nosym] : makesym-duplicate reverse edges, nosym-use original edge file
 - When the input graph is directed and you want to calculate directed RWR scores,
 then use 'nosym' in the 4th parameter.
 - When the input graph is directed and you want to calculate undirected RWR scores,
 then use 'makesym' in the 4th parameter.
 - When the input graph is undirected, use 'nosym' in the 4th parameter.

[block_width] : block width. usually set to 16.
[c] : mixing component. Default value is 0.85.

ex: ./run_rwrblk.sh rwr_edge rwr_query 16 3 nosym 8 0.85

The output of RWR-block is saved in the following HDFS directory:
rwr_vector :
 - Each line contains the RWR of each node in the format of
 (nodeid TAB "v"RWR_of_the_node).
 - For example, the line "1 v0.10231778333763829" means that the RWR of node 1 is 0.10231778333763829.
rwr_minmax : The minimum and the maximum RWR.

- The minimum RWR is the second column of the line that starts with "0".
- The maximum RWR is the second column of the line that starts with "1".

rwr_distr : The histogram of RWR. It divides the range of (min_RWR, max_RWR) into 1000 bins and shows the number of nodes which have RWRs that belong to such bins.

The working example of running RWR-block is in the do_rwrblk_catstar.sh.
If you run do_rwrblk_catstar.sh, it will run for 19 iterations until it converges.

3.6) Radius-plain

To run Radius-plain, you need to do the two things:

- copy the graph edge file to the HDFS directory hadi_edge
- execute ./run_hadi.sh

The edge file is a plain text file where each line is in SRC_ID TAB DST_ID format.
The range of node id is from 0 to number_of_nodes_in_graph - 1.
It is assumed that every node in the range exists.
Even if there are no edges to a node, it is assumed a one-node component.
Therefore, if the graph do not contain some nodes in the range, the calculated effective diameter might not reflect the effect diameter of the graph.
The effective radii and the maximum diameter are not affected by the missing nodes.

The syntax of run_hadi.sh is:

./run_hadi.sh [#_of_nodes] [#_of_reducers] [HDFS edge_file_path] [makesym or nosym] [enc or noenc]

[#_of_nodes] : number of nodes in the graph
[#_of_reducers] : number of reducers to use in hadoop.

- The number of reducers to use depends on the setting of the hadoop cluster.
- The rule of thumb is to use (number_of_machine * 2) as the number of reducers.

[HDFS edge_file_path] : HDFS directory where edge file is located
[makesym or nosym] : makesym-duplicate reverse edges, nosym-use original edge file

- When the input graph is directed and you want to calculate directed radii/diameter, then use 'nosym' in the 4th parameter.
- When the input graph is directed and you want to calculate undirected radii/diameter, then use 'makesym' in the 4th parameter.

  - When the input graph is undirected, use 'nosym' in the 4th parameter.
[enc or noenc] : use bit-shuffle encoding or not

```
ex: ./run_hadi.sh 16 3 hadi_edge makesym enc
```

The output of HADI-plain is saved in the following HDFS directory:
hadi_radius_<NODEID> :
  - Each line contains the radius of each node in the format of
    (nodeid   TAB   "bsf"maximum_radius:effective_radius).
  - For example, the line "1   bsf3:3" means that the maximum radius of node 1 is 3,
    and the effective radius of the node is also 3.
hadi_radius_summary_<NODEID> :
  - The first column is the radius.
  - The second column is the number of nodes with such radius.

The working example of running HADI-plain is in the do_hadi_catstar.sh.
If you run do_hadi_catstar.sh, it will run for 5 iterations.
In the end, it will output the following radii distributions:

```
Rad r    Count(r)
1        1
2        6
3        3
4        6
```

3.7) Radius-block

To run Radius-block, you need to do the two things:
  - copy the graph edge file to the HDFS directory hadi_edge
  - execute ./run_hadiblk.sh

The edge file is a plain text file where each line is in SRC_ID TAB DST_ID format.
The range of node id is from 0 to number_of_nodes_in_graph - 1.

The syntax of run_hadiblk.sh is:

./run_hadiblk.sh [#_of_nodes] [#_of_reducers] [HDFS edge_file_path] [makesym or nosym] [block_width] [enc or noenc]

[#_of_nodes] : number of nodes in the graph

[#_of_reducers] : number of reducers to use in hadoop.

- The number of reducers to use depends on the setting of the hadoop cluster.
- The rule of thumb is to use (number_of_machine * 2) as the number of reducers.

[HDFS edge_file_path] : HDFS directory where edge file is located

[makesym or nosym] : makesym-duplicate reverse edges, nosym-use original edge file

- When the input graph is directed and you want to calculate directed radii/diameter, then use 'nosym' in the 4th parameter.
- When the input graph is directed and you want to calculate undirected radii/diameter, then use 'makesym' in the 4th parameter.
- When the input graph is undirected, use 'nosym' in the 4th parameter.

[block_width] : block width. usually set to 16.

[enc or noenc] : use bit-shuffle encoding or not

```
ex: ./run_hadiblk.sh 16 3 hadi_edge makesym 2 noenc
```

The output of HADI-block is saved in the following HDFS directory:

hadi_radius_block_<NODEID> :

- Each line contains the radius of each node in the format of
  (nodeid   TAB   "bsf"maximum_radius:effective_radius).
- For example, the line "1   bsf3:3" means that the maximum radius of node 1 is 3, and the effective radius of the node is also 3.

hadi_radius_block_summary_<NODEID> : The distribution of radii of the graph

- The first column is the radius.
- The second column is the number of nodes with such radius.

The working example of running HADI-block is in the do_hadiblk_catstar.sh.

If you run do_hadiblk_catstar.sh, it will run for 5 iterations.

In the end, it will output the following radii distributions:

```
Rad r   Count(r)
1       1
2       6
3       3
```

```
4	6
```

3.8) Connected Component-plain

To run Connected Component-plain, you need to do the two things:
 - copy the graph edge file to the HDFS directory cc_edge
 - execute ./run_ccmpt.sh

The edge file is a plain text file where each line is in SRC_ID TAB DST_ID format.
The range of node id is from 0 to number_of_nodes_in_graph - 1.

The syntax of run_ccmpt.sh is:

./run_ccmpt.sh [#_of_nodes] [#_of_reducers] [HDFS edge_file_path]
[#_of_nodes] : number of nodes in the graph
[#_of_reducers] : number of reducers to use in hadoop.
 - The number of reducers to use depends on the setting of the hadoop cluster.
 - The rule of thumb is to use (number_of_machine * 2) as the number of reducers.
[HDFS edge_file_path] : HDFS directory where edge file is located

ex: ./run_ccmpt.sh 16 3 cc_edge

The output of HCC-plain is saved in the following HDFS directory:
concmpt_curbm :
 - Each line contains the component id of each node in the format of
   (nodeid	TAB	"msf"component_id_of_the_node).
 - For example, the line "2	msf1" means that the component id of node 2 is 1.
   The component id is the minimum node id of it.
concmpt_summaryout : The distribution of connected components.
 - The first column is the component id.
 - The second column is the number of nodes in the component.

The working example of running HCC-plain is in the do_ccmpt_catstar.sh.
If you run do_ccmpt_catstar.sh, it will run for 5 iterations until it converges.

3.9) Connected Component-block

To run Connected Component-block, you need to do the two things:
- copy the graph edge file to the HDFS directory cc_edge
- execute ./run_ccmptblk.sh

The edge file is a block text file where each line is in SRC_ID TAB DST_ID format.
The range of node id is from 0 to number_of_nodes_in_graph - 1.

The syntax of run_ccmptblk.sh is:

./run_ccmpt.sh [#_of_nodes] [#_of_reducers] [HDFS edge_file_path] [block_width]
[#_of_nodes] : number of nodes in the graph
[#_of_reducers] : number of reducers to use in hadoop.
- The number of reducers to use depends on the setting of the hadoop cluster.
- The rule of thumb is to use (number_of_machine * 2) as the number of reducers.

[HDFS edge_file_path] : HDFS directory where edge file is located
[block_width] : block width. usually set to 16.

ex: ./run_ccmptblk.sh 16 3 cc_edge 2

The output of HCC-block is saved in the following HDFS directory:
concmpt_curbm :
- Each line contains the component id of each node in the format of
  (nodeid TAB "msf"component_id_of_the_node).
- For example, the line "2 msf1" means that the component id of node 2 is 1.
  The component id is the minimum node id of it.

concmpt_summaryout : The distribution of connected components.
- The first column is the component id.
- The second column is the number of nodes in the component.

The working example of running HCC-block is in the do_ccmpt_catstar.sh.
If you run do_ccmpt_catstar.sh, it will run for 5 iterations until it converges.

4. Rebuilding Source Codes

PEGASUS distribution includes the source code. You can modify the code and rebuild the code. This section guides you how to rebuild the code.

1) List of Source Codes

Here is the list of source codes and their functions.

Source Code	Function
src/pegasus/degdist/	**Degree distribution**
- src/pegasus/degdist/DegDist.java	Degree distribution main class
src/pegasus/pagerank/	**PageRank**
- src/pegasus/pagerank/PagerankNaive.java	PageRank-plain main class
- src/pegasus/pagerank/PagerankBlock.java	PageRank-block main class
- src/pegasus/pagerank/PagerankInitVector.java	used in PageRank-block
- src/pegasus/pagerank/PagerankPrep.java	used in PageRank-block
src/pegasus/rwr/	**RWR**
- src/pegasus/rwr/RWRNaive.java	RWR-plain main class
- src/pegasus/rwr/RWRBlock.java	RWR-block main class
src/pegasus/hadi/	**Radius**
- src/pegasus/hadi/Hadi.java	HADI-plain main class
- src/pegasus/hadi/HadiBlock.java	HADI-block main class
- src/pegasus/hadi/HadiIVGen.java	used in HADI-block
src/pegasus/con_cmpth/	**Connected Component**
- src/pegasus/con_cmpth/ConCmpt.java	HCC-plain main class
- src/pegasus/con_cmpth/ConCmptBlock.java	HCC-block main class
- src/pegasus/con_cmpth/ConCmptIVGen.java	used in HCC-block
src/pegasus/matvec/	**Matrix-Vector Multiplication**
- src/pegasus/matvec/MatvecPrep.java	Convert plain data to block format
- src/pegasus/matvec/MatvecNaive.java	Naive matrix-vector multiplication
- src/pegasus/matvec/MatvecUtils.java	Utility function for matrix-vector
src/pegasus/pegasus/	**Pegasus Utils**
- src/pegasus/pegasus/PegasusUtils.java	Several utility functions
- src/pegasus/pegasus/GIMV.java	GIMV functions

src/pegasus/heigen/	**Linear Algebra**
- src/pegasus/heigen/NormalizeVector.java	Normalize vector
- src/pegasus/heigen/L1norm.java	L1 norm
- src/pegasus/heigen/L1normBlock.java	L1 norm using block method
- src/pegasus/heigen/Saxpy.java	Compute saxpy operation
- src/pegasus/heigen/SaxpyTextoutput.java	Compute saxpy operation
- src/pegasus/heigen/SaxpyBlock.java	Compute saxpy using block method
- src/pegasus/heigen/ScalarMult.java	Multiply a vector with a scalar
src/pegasus/column_joiner/	**Utility**
- src/pegasus/column_joiner/JoinTablePegasus.java	used in generating correlation plots

2) Building the Codes

Since the binary file pegasus-2.0.jar already exists, normally you don't need to build the code again. Thus, this is the instruction when you modify the source code and build it. Before building the code, you will need to specify the directory where hadoop-core.jar file is located.
Edit the build_pegasus.xml by finding the following line

```
<pathelement location="${hadoop.dir}/hadoop-${hadoop.version}-core.jar"/>
```

and change it to the directory where hadoop-core.jar file is located.
You can hard-code the path, or change other variables ${hadoop.dir} and ${hadoop.version}.
For example, to change to ${hadoop.dir} variable, edit the following line

```
<property name="hadoop.dir" value="${basedir}/../../../hadoop-${hadoop.version}"/>
```

and modify the value appropriately based on the hadoop installation directory.
The {basedir} means the current directory, and
the hadoop installation directory is the one that contains the 'hadoop-0.20.1-core.jar' file.
For example, suppose the PEGASUS and hadoop-0.20.1 are installed in the following directories.

```
PEGASUS:         /home/user/PEGASUS
Hadoop-0.20.1:   /home/user/hadoop-0.20.1
```

Then, the line should be changed to the following:

```
<property name="hadoop.dir" value="${basedir}/../hadoop-${hadoop.version}"/>
```

After editing the build_pegasus.xml file, build the code by executing build_pegasus.sh.