27 10 2016

SEWN 2016 labsheet 3 BSc version (assessed coursework)
Link analysis

Labsheets 3 and 4 are related. In this labsheet you will use the results of a crawl produced by a simple web crawler, i.e., a robot, to construct the link graph of the crawled web pages as an adjacency matrix, and calculate various statistics from this matrix. This will help you to understand how search engines build an index of the web. You will use the outputs of this labsheet in labsheet 4 to calculate PageRank for each page in the matrix.

Any BSc student attempting the MSc version of this lab may earn up to 10% additional credit.

The maximum grade achievable for this labsheet is 100%.

Part 1: Creating the adjacency matrix

1. The first requirement of this labsheet is to convert the crawl's results into an adjacency matrix. There are approximately 500 visited pages in this data and the links between them and other non-visited pages. You can use one of the following options to obtain the data:[1]

   a. Download file:

      http://www.dcs.bbk.ac.uk/~martin/sewn/ls3/sewn_2016_labsheet_3_full_crawl.txt

      Which lists the raw results of the crawl in the following format:

```
Visited: <URL of page A>
   Link: <URL of page B>
   Link: <URL of page C>
Visited: <URL of page B>
   Link: <URL of page C>
Visited: <URL of page C>
   Link: <URL of page A>
...
```

   b. Download file:

      http://www.dcs.bbk.ac.uk/~martin/sewn/ls3/sewn_2016_labsheet_3_full_crawl.sql

      Which represents the results of the crawl as a MySQL database. See Part 3(a) for further information.

   c. Download file:

      http://www.dcs.bbk.ac.uk/~martin/sewn/ls3/sewn_2016_labsheet_3_full_crawl.xlsx

      Which represents the results of the crawl in a Microsoft Excel spreadsheet which duplicates the database arrangements from (b) above. One sheet of the spreadsheet is for visited pages and another for links.

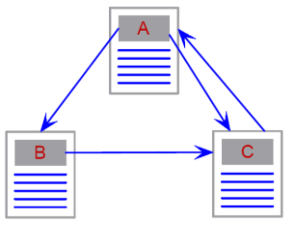
   d. As an alternative to options (a) and (b) and (c) above, you are allowed to use option (d) which consists of a *BlueJ* Java project at:

      http://www.dcs.bbk.ac.uk/~martin/sewn/ls3/sewn_2016_labsheet_3_bsc_java_projcode.zip

      Which creates an in-memory data structure to represent the crawl's results. See Part 3(b) for further information.

[1] Minor differences in the crawl's results may exist between the various approaches listed.

Whichever option you decide upon, the crawl's results represents a link graph, i.e. where A, B and C are visited pages, as displayed below.



2. Now that you have the crawl's results, we require you to represent them electronically as an *adjacency matrix*. You are free to use any method to do this, e.g. you can use a program to store the matrix in computer memory, write it other database table(s) or spreadsheet, but this method must be demonstrable in the PC laboratories at the college.

   An adjacency matrix is a labelled graph is with rows and columns labelled by graph vertices, with a 1 or 0 in position according to whether the vertices are adjacent or not. The link graph above becomes the following adjacency matrix:

```
             (inlinks)
              | A B C
             --------
             A| 0 1 1
(outlinks)   B| 0 0 1
             C| 1 0 0
```

   Where the number of outlinks from a page can be seen by reading horizontally and the inlinks to a page by reading vertically. Further information on adjacency matrices can be found at:

   http://mathworld.wolfram.com/AdjacencyMatrix.html

   You should be aware that any crawl of the web is *partial*, i.e. all crawls, including those carried out by commercial search engines, are partial given the size of the WWW. This means that a page may have several links to another, and that there are also *dangling* links to pages not visited during the crawl. When populating your adjacency matrix for option (a), (b) (c) or (d), you must decide how to deal with those pages listed as links, but not visited. Moreover, many of the links between pages are relative and will need to be resolved.

   How you address these issues will influence the remainder of the work you undertake for this labsheet and eventually labsheet 4.

   Options (a), (b) or (c) carry extra credit.

   For testing purposes, you may want to use the following basic crawl file to create test matrices:

   http://www.dcs.bbk.ac.uk/~martin/sewn/ls3/sewn_2016_labsheet_3_basic_crawl.txt

   This basic crawl consists of approximately a dozen pages and the links between them.

Part 2: Link analysis statistics

Once you have created the adjacency matrix, use it to compute the following statistics for link analysis:

a. The number of inlinks to each page.
b. The number of outlinks from each page.
c. The average (mean) number of inlinks to a page, variance and the standard deviation.
d. The average (mean) number of outlinks from a page, variance and the standard deviation.
e. The *degree distribution* of inlinks and outlinks, i.e. tables such as the following:

# outlinks	# pages
1	*p*
2	*q*
…	*r*

# inlinks	# pages
1	*x*
2	*y*
…	*z*

The outlinks table should list the number of pages *p* with 1 outlink, number of pages *q* with 2 outlinks and so on. The statistics above can be written to web page, screen or files(s), but you need to include the results as part of your submission.

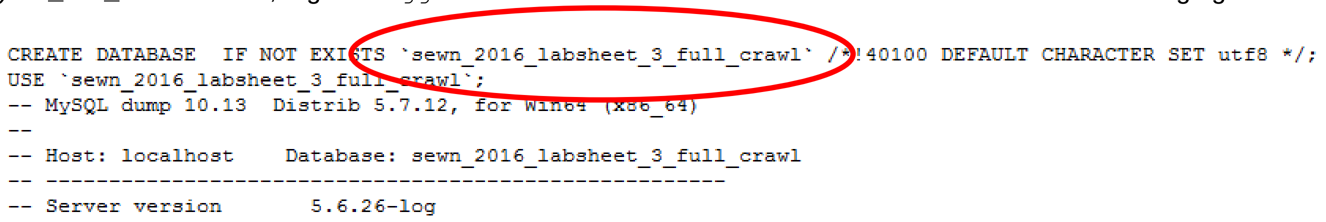
Part 3: Additional information

a. **MySQL database:** The MySQL database of the crawl's results referred to in Part 1(b) consists of two tables forming a one-to-many relationship, i.e.:

```
CREATE TABLE `VisitedPage` (
  `VisitedPageNo` int(11) NOT NULL AUTO_INCREMENT,
  `VisitedPageURL` varchar(512) NOT NULL,
  PRIMARY KEY (`VisitedPageNo`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `Link` (
  `LinkNo` int(11) NOT NULL AUTO_INCREMENT,
  `LinkURL` varchar(512) NOT NULL,
  `VisitedPageNo` int(11) NOT NULL,
  PRIMARY KEY (`LinkNo`),
  KEY `VisitedPageNo` (`VisitedPageNo`),
  CONSTRAINT `linkFK` FOREIGN KEY (`VisitedPageNo`) REFERENCES `VisitedPage`
(`VisitedPageNo`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

In order to create and populate the database on the college's MySQL server with the crawl's results, you will need to: (1) download the file from the link given in Part 1(b), and (2) replace every instance of name `sewn_2016_labsheet_3_full_crawl` in the file with the name of your college MySQL database, i.e. *your_dcs_usernamedb*, e.g. `fbloggs01db`. There are at least three instances which will need changing.



```
CREATE DATABASE  IF NOT EXISTS `sewn_2016_labsheet_3_full_crawl` /*!40100 DEFAULT CHARACTER SET utf8 */;
USE `sewn_2016_labsheet_3_full_crawl`;
-- MySQL dump 10.13  Distrib 5.7.12, for Win64 (x86_64)
--
-- Host: localhost    Database: sewn_2016_labsheet_3_full_crawl
-- ------------------------------------------------------
-- Server version	5.6.26-log
```

File `sewn_2016_labsheet_3_full_crawl.sql` can now be opened in MySQL Workbench and executed.

If you do not have access to the college's MySQL server, please contact Systems Group.

b. **BlueJ Java project:** The BlueJ project referred to in Part 1(d) is called `SEWN_2016_Labsheet_3_Full_Crawl` and it consists of three Java classes:

1. **`VisitedPage`:** This class defines an object representing a visited page. There are two attributes, i.e. `visitedPageURL` is a `String` for the page's URL and `links` is an `ArrayList` of type `String` to store the links from that page.
2. **`VisitedPage_ReaderWriter`**: This class is used for writing the crawl data to an `ArrayList` of type `VisitedPage`. The class also has a `String` attribute called `path` to store the location of the file storing the crawl's results downloaded in Part 1(a).
3. **`VisitedPage_Main`**: This is the *executable* class with a `main` method which will create a single instance of class `VisitedPage_ReaderWriter` to read, store and display the crawl's results. You may need to change the value of the `path` variable depending upon the location of the file `sewn_2016_labsheet_3_full_crawl.txt` storing the crawl's results.

No further documentation for this project is available and no support will be provided for either it or MySQL.

What to hand in:
Submit a single .zip file to the **Labsheet 3 Link analysis** drop box in Moodle containing:

1. The crawl data for Part 1.
2. Any program code, or equivalent implementation, for Part 1 should be supplied together with any special instructions for compiling and running your solution (if there are several code files, please include the folder structure in your `.zip` file). Depending upon your solution, you may be required to demonstrate it in one of the PC laboratories at the college.
3. An A4 report of no more than two pages (`.doc(x)` or `.pdf` format) describing how your solution to Part 1 works and also why you chose the option used. The report should also include a brief description of how your solution resolves relative and duplicate links and how you have handled *dangling* links when populating your adjacency matrix.
4. The degree distribution and statistical data from Part 2.
5. Any references to books or on-line material for this labsheet should be included in your report.

Submission deadline: 17 11 2016.

Late assignments: No extensions are available as for this lab and any late submissions will be graded as per the guidelines of the relevant course being studied.