Istat SW for webscraping

Donato Summa

THE CONTRACTOR IS ACTING UNDER A FRAMEWORK CONTRACT CONCLUDED WITH THE COMMISSION

Shortly we have 2 use cases

- Url retrieval
- Webscraping of enterprise websites

Use case 1

Url retrieval

URLs retrieval

- In order to be able to crawl and scrape content on firm websites you need first of all to know the list of that website addresses

- If you don’t have the list of that website addresses available you have to procure it

- We tried several traditional ways in order to acquire the full list without success

What we have

- A list with about 200K enterprise records containing several information:
  - ISTAT code (primary key)
  - Official name of the enterprise
  - Certified mail
  - VAT
  - City
  - Province
  - Telephone number

The automated URL retrieval procedure

- Idea: obtain an enterprise web site (if it exists) starting from a search of the enterprise name in a search engine

The automated URL retrieval procedure

1. Input from existing sources (Register plus other archives): denomination, address, telephone number, fiscal code, …)

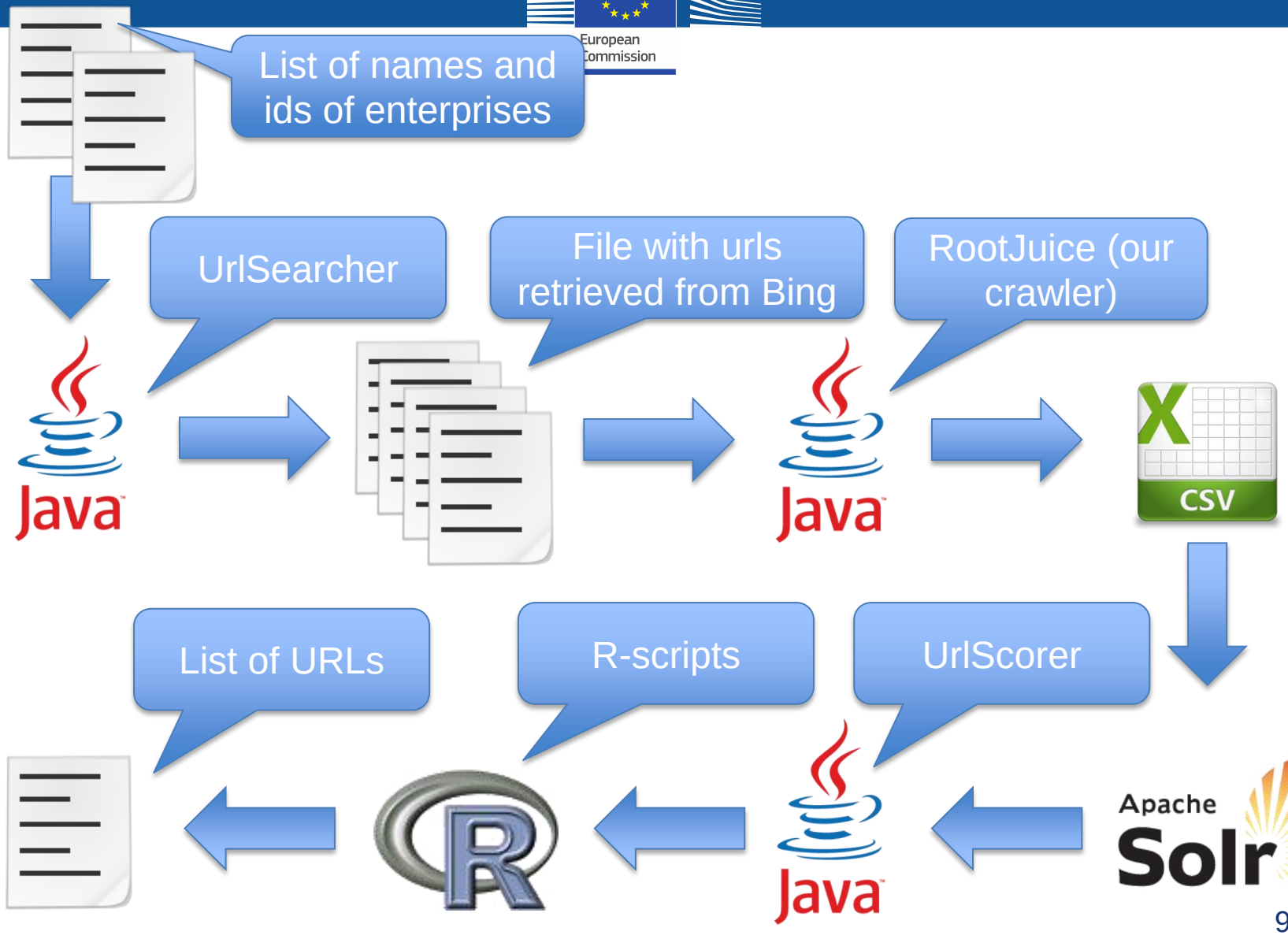
2. For each enterprise in the target population:
   a) Introduce the denomination into a search engine
   b) Obtain a list of the first k resulting web pages
   c) For each one of these results, calculate the value of binary indicators. For instance:
      - the URL contains the denomination (Yes/No);
      - the scraped website contains geographical information coincident with already available in the Register (Yes/No);
      - the scraped website contains the same fiscal code in the Register (Yes/No);
      - the scraped website contains the same telephone number in the Register (Yes/No);
      - …
   d) Compute a score on the basis of the values of the above indicators.

The automated URL retrieval procedure

3. On the subset of enteprises for which the URL is known (training set), model the relation between the binary indicators plus the score, and the success/failure of the found URL

4. Apply the model to the subset of enterprises for which the URL is not known, in order to decide if the found URL is acceptable or not.

Url retrieval flow



Step 1 - UrlSearcher

- takes in input 2 files containing the list of the firm names and the corresponding list of firm IDs.
- for each enterprise the program queries a search engine (we are actually using Bing) and retrieves the list of the first 10 urls provided by the search engine
- these urls are then stored in a file so we will have one file for each firm
- at the end, the program reads each produced file and creates the seed file

Step 1 - UrlSearcher

- The seed file is a txt file within wich every row is so composed:

url_retrieved + TAB +
firm_Id + TAB +
position_of_the_url

eg: www.mywebsite.com/aboutus 111 3

Step 2 – RootJuice (our crawler)

- takes in input 3 files:
  - the seed file just produced
  - a list of url domains to avoid (directories domains)
  - a configuration file
- for each row of the seed file (if the url is not in the list of the domains to avoid) tries to acquire the HTML page
- from each acquired HTML page the program extracts just the textual content of the fields we are interested in and write a line in a CSV file

Step 2 – RootJuice (our crawler)

- The structure of each row is this:

id + TAB + **url** + TAB + **imgsrc** + TAB + **imgalt** + TAB +

links + TAB + **ahref** + TAB + **aalt** + TAB + **inputvalue** + TAB +

inputname + TAB + **metatagDescription** + TAB +

metatagKeywords + TAB + **firmId** + TAB + **sitoAzienda** + TAB +

link_position + TAB + **title** + TAB + **text_of_the_pagebody**

Step 3 – Load scraped data into Solr

- The produced solrInput.csv file is the input for the next step of the process

- Now that we have the scraped textual content of the html pages we need to index/persist it in a enterprise search platform (Solr)

Step 4 – UrlScorer

- It requires in input 2 parameters :
  - a file containing information about each firm
  - the Solr index directory

- For each document in Solr the program calculates a score vector and assigns a score on the basis of predefined conditions (eg: in the text of the document we can find the vat code or the telephone number)

Step 4 – UrlScorer

- Finally it writes down a row in a CSV file with the following structure:

  firmId + TAB +
  linkPosition + TAB +
  url + TAB +
  scoreVector + TAB
  score

Step 5 – UrlMatchTableGenerator

- The CSV file obtained in the step 4 is the input for the machine learning program that will try to predict the correct url for each firm.

- In order to be able to accomplish this task you have to instruct the learner in advance providing it a test set (a similar CSV file containing the extra boolean column “is the found url correct").

- We created this test set using a custom Java program (UrlMatchTableGenerator) that merges the CSV file from step 4 with a list of correct sites

Step 5 – UrlMatchTableGenerator

- Once the learner will be instructed it will be able to "recognize" the correct url for a firm without knowing the real official site in advance (as in the train phase)

Obtained results

- For each firm considered in the test set we compared the most probable official URL found by the procedure with the official URL that we already knew.
- In particular we compared only the domain part of the url (eg. www.**rossi.it**/aboutUs)
- We foud exact match in 64% of the cases

Considerations

The real success percentage is probably bigger than obtained because sometimes the official web site that we know is not correct because:

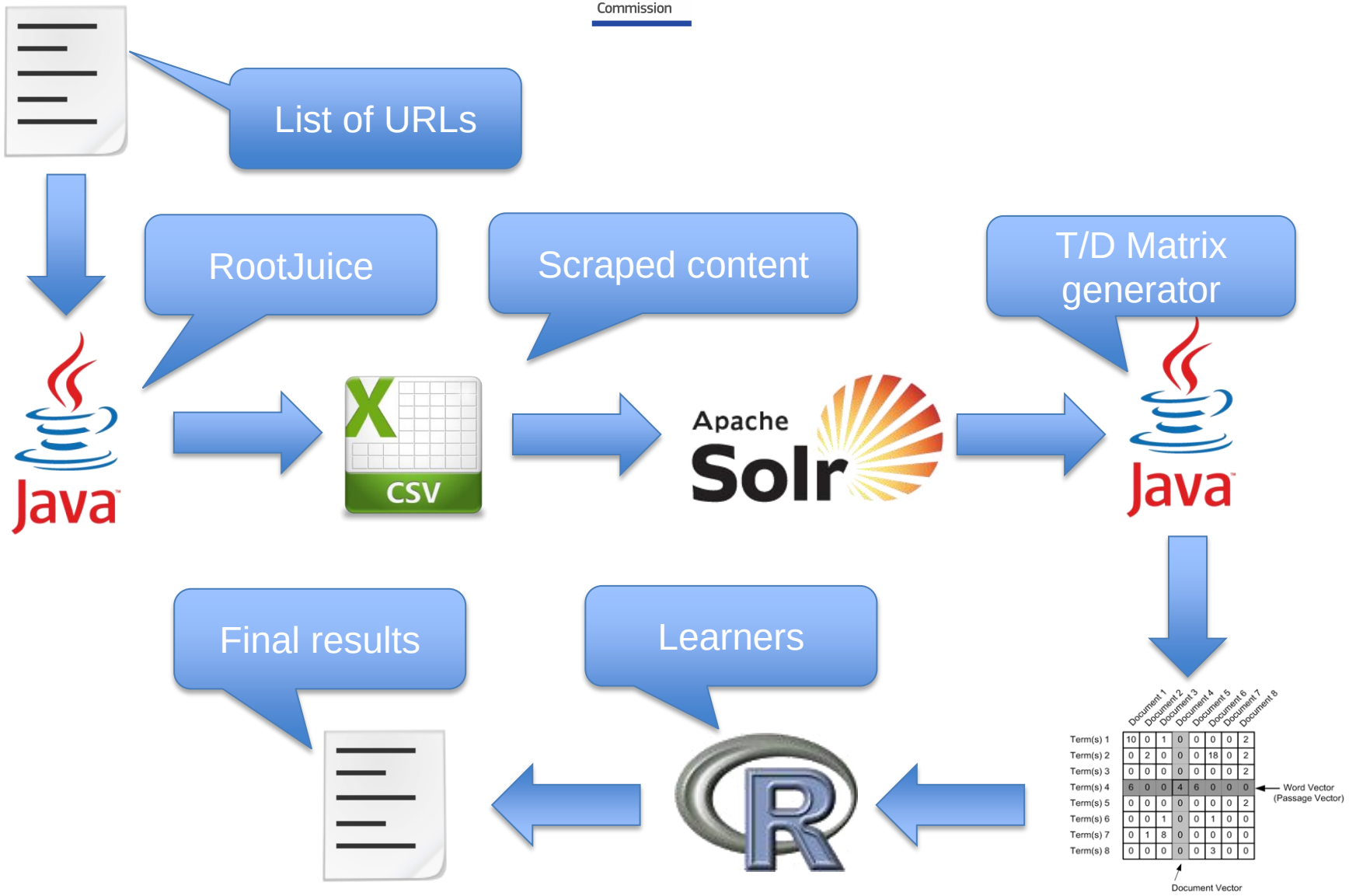
- it is outdated (the url is changed)
- it does not exist anymore
- wrong domain (e.g. "rossi.it" instead of "rossi.com")
- it is only an information page with contacts (paginegialle)
- it is the site that sells the products of the enterprise (e.g. mediaworld.it)
- it is the site of the mother company (franchising enterprises)

Probably in this cases Bing and our algorithm find the correct site but we consider it uncorrect because different from the one we know

Use case 2

Web scraping of enterprise websites

Web scraping flow



Step 1 – RootJuice (our crawler)

- As we already saw it takes in input 3 files:
  - the seed file just produced
  - a list of url domains to avoid (directories domains)
  - a configuration file
- for each row of the seed file (if the url is not in the list of the domains to avoid) tries to acquire the HTML page
- from each acquired HTML page the program extracts just the textual content of the fields we are interested in and write a line in a CSV file

Step 2 – Load scraped data into Solr

- The produced solrInput.csv file is the input for the next step of the process
- Now that we have the scraped textual content of the html pages we need to index/persist it in Solr

Step 3 – FirmsDocTermMatrixGenerator

- It takes in input 2 parameters :
  - the folder path of Solr index containing our data
  - the folder path within which the output matrix will be saved

- The output will be a matrix having :
  - on the first column all the relevant stemmed terms found in all the documents
  - on the first row all the firms id contained in the storage platform
  - each cell will contain the number of occurencies of the specific term in all the documents referring the specific firm

European Commission

Thank you for your attention !