Online Based Enterprise Characteristics Experimental Statistics: Italy

Aims
The general objective is to infer some enterprises characteristics by accessing their websites through Web scraping techniques. We, as Italy, have taken part to this experimental statistics in the first ESSnet Big Data project. In particular we got some experience in the following use cases:

(i) URL inventory (The objective is to identify, if it exists, the most probable official website for a set of enterprises)
(ii) eCommerce (Given a set of enterprises’ websites, the objective is to detect whether the websites provide or not web sales facilities.)
(iii) Social media presence (Given a set of enterprises’ websites, the objective is to detect for each website the presence of the enterprise on the social media.)
(iv) Job advertisement (Given a set of enterprises’ websites, the objective is to detect whether the websites provide or not job advertisements.)

Predicted values can be used for a twofold purpose: (i) at unit level, to enrich the information contained in the register of the population of interest; (ii) at population level, to produce estimates. In order to reach this objectives, the work performed had to face methodological, technological and legal challenges. Legal issues and possible solutions has been pointed out in one of the deliverables of the first ESSnet Big Data project (deliverable 2.11). The outcomes produced were experimental statistics available on the ESSnet Big Data 1 project wiki1.

Data Sources
Several data sources were used for different purposes at different stages of the experimentation. It is worthwhile to underline the fact that the used data sources can be divided in two main categories: (i) initial data sources immediately available at the beginning of the process; (ii) intermediate data sources (obtained as an output of a previous step such as the URL inventory use case).

In the URL inventory use case the used data sources have been:

- an initial set of enterprises having at least 10 employees obtained from existing sources (Statistical Business Register, ICT survey and some administrative sources such as Consodata). For each enterprise we had several information: denomination, address, telephone number, fiscal code, etc. It is important to notice that just for a subset of enterprises was also available the official website.

---

1 https://webgate.ec.europa.eu/fpfis/mwikis/essnetbigdata/index.php/Main_Page1
- a list of ten URLs (on average) for each considered enterprise obtained as a result of a search on a Web search engine
- web scraped textual data coming from the homepages of the enterprises official websites

In the remaining use cases (eCommerce, social media presence and job advertisement) the used data sources have been:

- the list of the official URLs of the enterprises on subject (partially available since the beginning and partially obtained as output from the URL inventory use case)
- the corresponding list of records related to the considered enterprises (Statistical Business Register)
- web scraped textual data coming from the enterprises official websites

**Methodology**

From a conceptual point of view it was designed a generic reference logical architecture made of several building blocks organized into four main layers (Internet access, Storage, Data preparation, Analysis) as shown in Figure 1:

![Generic reference logical architecture](image)

This reference logical architecture was then properly instantiated with concrete software tools in order to fulfill the use cases requirements and objectives. There were prepared two specific configurations, one for the URL inventory use case (Figure 2) and a second one for the remaining use cases (Figure 3).
Figure 2 - Specific implementation for the URLs inventory use case

Figure 3 - Specific implementation for all other use cases
URL Inventory use case

The objective of this pilot consists of identifying (if it exists) the most probable official website for a set of enterprises (with an associated set of identifiers such as the denomination, the fiscal code, the economic activity, etc.) by using a semi-automated procedure. In order to obtain a list of URLs for a particular enterprise its name was searched using a search engine and the obtained URLs were scraped, stored and analysed. In the analysis phase the list of enterprises with a website known in advance were used as a training set for the learners, while the remaining enterprises and their associated URLs found by the procedure were used as a test set.

Pipeline stages

Our input was a set of enterprises having at least 10 employees obtained from existing sources (Business Register and some administrative sources); for each enterprise we had several information: denomination, address, telephone number, fiscal code, etc.

We passed the list of enterprises as input to a program named UrlSearcher that for each enterprise contained in the list:

- Introduce the denomination into a search engine (we used Bing)
- Obtain a list of the first 10 resulting web pages (URLs)
- Print the obtained web addresses in a file usually named seed.txt

We passed the seed.txt as input to our webscraper called RootJuice that retrieves the textual content of each URL and prints it on a CSV file that will be loaded into a storage platform named Solr.

Once we had the scraped information stored in Solr as documents (one Solr document per URL) we launched UrlScorer that reads these documents and assigns to each of them a score on the basis of the values of some 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);
- ...

On the subset of enterprises for which the URL was known (training set), we used custom Java SW and custom R scripts in order to model the relation between the binary indicators plus the score, and the success/failure of the found URL. At the end we applied the model to the subset of enterprises for which the URL is not known, in order to decide if an automatically found URL was acceptable or not.

Functional description of each block

UrlSearcher\(^2\) is a custom Java application that takes as input a list of enterprises names and identification numbers and, for each of them, performs a query to a search engine and produces a text file containing the first 10 URLs returned by the search engine. We used this program in order to collect a list of websites for a given enterprise name. The underlying assumption is that, if an

\(^2\) https://github.com/SummaIstat/UrlSearcher
enterprise has an official website, this should be found within the first 10 results provided by a search engine.

RootJuice is a custom Java application that takes as input a list of URLs and, on the basis of some configurable parameters, retrieves the textual content of that URLs and prints it on a file that will be loaded into a storage platform named Solr.

Apache Solr is a NoSQL database. It parses, indexes, stores and allows searching on scraped content. Providing distributed search and index replication, Solr is highly scalable and, for this reason, suitable to be used in Big Data context.

UrlScorer is a custom Java program that reads one by one all the documents contained in a specified Solr collection and assigns to each of them a score on the basis of the values of some indicators. In particular it calculates 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);
- etc.

Custom R scripts are used in the analysis phase, which is the last phase of the process.

In our case study, our input training dataset consisted of 81912, of which 73006 records had at least one page fetched. On the basis of the output scoring dataset we first associated to each enterprise of the 73006 sized set the link with the highest score. As we knew if the link was correct or not, a dichotomous variable correct_Yes_No said if the URL was the right one or not: this variable plays the role of the Y variable, to be predicted by the model. Together with this information, variables indicating success or failure of the search of telephone, VAT code, municipality, province and zip code play the role of the X variables (predictors), together with the link position and coincidence of the central part of the URL with the name of the enterprise (simple URL). This initial set was split into two equal size subsets, the first acting as the proper training set to fit the model, the second as the test set used to evaluate the performance of the model. Different learners have been fitted (also by using the Java custom program UrlMatchTableGenerator as support) and evaluated, namely Neural Networks, Random Forest and Logistic Model. Their performance has been evaluated by considering the classic indicators, that is accuracy, sensitivity, specificity and F-measure (harmonic mean of recall and precision). Taking into account the statistical properties of the logistic model, this learner has therefore been preferred to the others, also because of the interpretation of the score as a probability.

Once applied to the test set, units have been sorted in ascending order with respect to the score assigned by the logistic model, and have been grouped in 10 balanced classes. By taking all the links in a given class, the error rate depends on the number of false positives in that class. It is clear that the error rate decreases as the score (i.e. the probability of correct link) increases. If the acceptation threshold value is set to of 0.573 as the one to decide if a link is correct or not, the upper five classes are accepted in toto and the mean error that can be expected is 0.13, and the total recall is 0.75. In other words, 75% of correct links can be found, together with 13% or erroneous ones.

---

3 https://github.com/Summalstat/RootJuice
4 https://lucene.apache.org/solr/
5 https://github.com/Summalstat/UrlScorer
6 https://github.com/Summalstat/UrlMatchTableGenerator
If the refusal threshold value is set to 0.507, the lower five classes are discarded, losing in this 23% of correct links. It is possible to choose the 5th class (containing about 9.5% of cases), where true and false positives are balanced, as the one to be controlled interactively.

E-Commerce, social media presence and job advertisement use cases

Given a set of enterprises’ websites, the pilots had the objective to detect:

- whether the websites provide or not web sales facilities;
- for each website the presence of the enterprise on the social media;
- whether the websites provide or not job advertisements.

Pipeline stages

For all the use cases the process followed was the same, the only difference was contained in the final phase of analysis within which the custom R script was slightly changed in accordance with the specific objectives to be reached.

Our input was a txt file (named seed.txt) containing the list of enterprises’ websites specified as URLs. The following steps were executed:

- Seed.txt is taken as input by RootJuice together with a configuration file and a list of URLs to filter out.
- RootJuice scrapes the textual content of the websites provided as input and writes the scraped content to a CSV file.
- The CSV file is uploaded to Solr (via command line or via application programs).
- Within Solr, during the loading phase, some data preparation steps are performed, namely: lowercasing, removal of stopwords.
- An ad-hoc Java program performs tokenization and lemmatization in two languages.
- An ad-hoc Java program generates a Term document matrix with one word for each column and one enterprise for each row and containing the occurrences of the word in the set of webpages related to the website in the corresponding row.
- The resulting matrix is provided as input for the analysis step.
- The analysis step consists in taking the subset of enterprises answering to the 2016 ICT survey and considering this subset as the ground truth for fitting and evaluating different models ("Support Vector Machines", "Random Forest", "Logistic", "Boosting", "Neural Net", "Bagging", "Naive Bayes") by performing:
  - Features selection, obtained by sequentially applying
    - Correspondence Analysis (reduction from about 50,000 terms to 1000 terms)
    - Importance in generating Random Forests (from 1000 terms to 200 terms)
  - Partitioning of data in a train and test sets equally balanced
  - Model fitting on the train set and evaluation on the test set

The evaluation has been carried out by considering different indicators, mainly accuracy and F1-measure. Two of these models, Logistic and Random Forest, have been used to predict the values of all enterprises for which the scraping was successful.

Functional description of each block
RootJuice is a custom Java application that takes as input a list of URLs and, on the basis of some configurable parameters, retrieves the textual content of that URLs and prints it on a file that will be loaded into a storage platform named Solr.

Apache Solr is a NoSQL database. It parses, indexes, stores and allows searching on scraped content. Providing distributed search and index replication, Solr is highly scalable and, for this reason, suitable to be used in Big Data context.

FirmDocTermMatrixGenerator\(^7\) is a custom Java application that reads all the documents (related to scraped enterprises' websites) contained in a specified Solr collection, extracts all the words from them and generates a matrix having: (i) one word for each column, (ii) one enterprise for each row and (iii) the number of occurrences of each word in each firm set of scraped webpages in the cells.

Custom R Scripts have been developed:
- Freqs.R, CA_words_selection.R, select.R to perform the feature selection by applying Correspondence Analysis
- randomForest.R to perform the feature selection by applying importance in Random Forest generation
- predictions.R to fit models on train dataset and evaluate them on test dataset
- compute_estimates.R to apply fitted models to the total number of enterprises for which the scraping was successful, calculate estimates for different domains, and compare to sampling estimates
- compute_variance_logistic.R and compute_variance_RF.R to calculate model variance for estimates produced by applying logistic model and RandomForest model.

Results

At the end of the experimentation the following outputs were produced:

- Experimental statistics with methodological notes:
  - URL retrieval\(^8\)
  - Ecommerce\(^9\)
  - Job advertisements\(^10\)
  - Social media presence\(^11\)
  - Estimates of the modalities of use of websites by enterprises\(^12\)
- Project deliverables\(^13\)
- Technical notes\(^14\)

---
\(^{7}\) https://github.com/SummaIstat/FirmDocTermMatrixGenerator
\(^{8}\) https://webgate.ec.europa.eu/FPFIS/Mwikis/Essnetbigdata/index.php/WP2_URL_retrieval1
\(^{9}\) https://webgate.ec.europa.eu/FPFIS/Mwikis/Essnetbigdata/index.php/WP2_Ecommerce1
\(^{10}\) https://webgate.ec.europa.eu/FPFIS/Mwikis/Essnetbigdata/index.php/WP2_Job_advertisements1
\(^{11}\) https://webgate.ec.europa.eu/FPFIS/Mwikis/Essnetbigdata/index.php/WP2_Social_media_presence1
\(^{12}\) https://www.istat.it/en/archive/216641
\(^{13}\) https://webgate.ec.europa.eu/FPFIS/Mwikis/Essnetbigdata/index.php/WP2_Reports,_milestones_and_deliverables1
\(^{14}\) https://webgate.ec.europa.eu/FPFIS/Mwikis/Essnetbigdata/index.php/WP2_Working_area1