**ESSnet Big Data**

**Specific Grant Agreement No 1 (SGA-1)**


Framework Partnership Agreement Number **11104.2015.006-2015.720**  
Specific Grant Agreement Number **11104.2015.007-2016.085**

**Work Package 2**  
**Web scraping Enterprise Characteristics**

**Deliverable 2.2**  
**Methodological and IT Issues and Solutions**  
**Version 2017-07-31**

**Prepared by:**

Galia Stateva (BNSI, Bulgaria)  
Olav ten Bosch (CBS, Netherlands)  
Jacek Maślankowski (GUS, Poland)  
Giulio Barcaroli, Monica Scannapieco, Donato Summa (ISTAT, Italy)  
Matthew Greenaway (ONS, United Kingdom)  
Ingegerd Jansson, Dan Wu (SCB, Sweden)

ESSnet co-ordinator:

Peter Struijs (CBS, Netherlands)  
[p.struijs@CBS.nl](mailto:p.struijs@CBS.nl)  
telephone : +31 45 570 7441  
mobile phone : +31 6 5248 7775
# Table of contents

Table of contents............................................................................................................................................... 1

1 Executive Summary .......................................................................................................................................... 2

2 General Motivations for Web Scraping of Enterprises Web sites ................................................................. 4

3 Description of a Reference Framework for Web Scraping of Enterprises’ Websites ....................................... 6

4 Methodological Issues and Solutions ............................................................................................................... 9
  
  4.1 Specific vs generic scraping: when/what ............................................................................................... 9
  4.2 Analysis techniques: machine learning vs deterministic approaches ...................................................... 10
  4.3 Review of methods used in pilots ............................................................................................................ 11
  4.4 Potential use of pilots’ output ................................................................................................................. 14
  4.5 Conclusions ............................................................................................................................................. 15

5 Technological Issues and Solutions ............................................................................................................. 16
  
  5.1 Review of the technological environments used in pilots ..................................................................... 16
  5.2 Issues and Solutions ............................................................................................................................... 21
  5.3 Lessons Learnt and Conclusions .......................................................................................................... 22

6 Future Challenges in Web-scraping Enterprise Websites ............................................................................. 23

7 Appendixes by Use Cases ............................................................................................................................... 25
  
  7.1 Use Case 1 - URL Inventory .................................................................................................................. 25
  7.2 List of Pilots: Use Case 1 ....................................................................................................................... 27
  7.3 Use Case 2 - ECommerce ..................................................................................................................... 48
  7.4 List of Pilots Use Case 2 ....................................................................................................................... 50
  7.5 Use Case 3 – Job Advertisements ......................................................................................................... 61
  7.6 List of Pilots Use Case 3 ....................................................................................................................... 62
  7.7 Use Case 4 – Social media presence on enterprises’ web pages .......................................................... 72
  7.8 List of Pilots Use Case 4 ....................................................................................................................... 73
1 Executive Summary
Monica Scannapieco (Istat)

This document describes the technical work performed within the workpackage “Web Scraping Enterprise Characteristics” (WP2) of the ESSnet Big Data SGA1. The general objective of this workpackage was to infer some enterprises characteristics by accessing their websites through Web scraping techniques.

In order to reach this objective, the work performed within WP2 had to face methodological, technological and legal challenges. In the deliverable 2. 1\(^1\), legal issues and possible solutions were discussed in detail. In this document, instead, the main focus will be on methodological and technological aspects.

The work was organized according to four principal phases, namely:

- **Phase 1**: Specification of a set of use cases to define the detailed scope of the work.
- **Phase 2**: Design of one or more pilots for each use case to be implemented by the different countries participating to WP2 (i.e. Bulgaria, Netherlands, Poland, Italy, United Kingdom and Sweden).
- **Phase 3**: Pilot implementations by the different countries.
- **Phase 4**: Identification of the main methodological and technological issues and solutions faced within the pilots.

Some figures on the performed work are:

- **Six different use cases** were identified.
- **Four use cases** out of the six were selected to be demonstrated by specific pilots.
- **Sixteen different pilots** were implemented, namely: six for use case 1 (with Bulgaria implementing two pilots with two different technologies), four for use case 2, three for use case 3 and three for use case 4.

The main findings can be summarized as follows:

- The complex pipeline for processing data scraped from enterprises’ websites has been defined in detail and shared among the participants. This pipeline can be considered as a reference one to which mapping specific technological and methodological choices. A set of logical building blocks have been identified for each phase of the pipeline.
- From a methodological perspective, both deterministic and machine learning methods were used in the pilots. On one side, we learned that even with different methods good results can be achieved. On the other side, however, we saw that in some cases there can be a convergence of methods (e.g. the URL retrieval pilot where Italy, Bulgaria and the Netherlands applied the same methodology). 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. The issue of measuring the quality of data pertaining the unit level has been faced in the piloting phase. In particular, for

\(^1\) Available at: https://webgate.ec.europa.eu/fpfis/mwikis/essnetbigdata/images/a/a0/WP2_Deliverable_2_1_15_02_2017.pdf
instance, when employing machine learning methods the quality can be measured by considering the same indicators produced to evaluate the model fitted in the training set. Under given conditions (if the training set is representative of the whole population), the measure of the accuracy (and also of other indicators like sensitivity and specificity) calculated in this subset can be considered as a good estimate of the overall accuracy. The issue of measuring the quality of population estimates making use of predicted values has also been focused. However, specific solutions to that are still under investigation.

- From an IT perspective, performance is a key issue especially when downloading and processing the whole websites. Processing unstructured information is very CPU and memory consuming, especially with machine learning algorithms, and as a result not very efficient. A sustainability issue is also very relevant, due to the fact that big data tools are changing very frequently as well as the website technology, there is a need to provide an agile-like development of tools. For the storage, the possible choices are between filesystem (CSV, JSON etc.), NoSQL database (Solr, Cassandra, Hbase etc.) or relational database (MySQL, PostgreSQL, SQL Server etc.). The decision of the use of particular data storage should be taken according to the volume of the data and the type of data to be stored. Finally, although the frameworks are developed in particular countries, it is possible to apply them in other countries as well without any major changes. For instance URLSearcher developed by Istat was tested on Bulgarian and Polish websites as well.

Some indicators can be computed as outputs of the developed pilots, and can be considered as experimental statistics. These include:

- Rate(s) of retrieved URLs from an enterprises’ list.
- Rate(s) of enterprises engaged in ecommerce from enterprises websites.
- Rate(s) of enterprises that have job advertisements on their websites.
- Social media presence, in terms of both (i) Rate(s) of enterprises that are present on social media from their websites and (ii) Percentage of enterprises using Twitter for a specific purpose.

The organization of this document follows the sequence of phases introduced at the beginning of this section, with the only exception that we will describe Phase 4 before Phase 3, whose details are provided as an appendix, for the sake of readability. In particular:

- Section 2 of this document describes the results of Phase 1, both in terms of identified use cases and candidatures by involved countries to implement them.
- Section 3 deals with generic building blocks identified as reference ones to guide the implementation of the different pilots. It details Phase 2.
- Section 4 details Phase 4 with respect to methodological challenges, including:
  - Generic and specific web scraping approaches and their possible usages.
  - Deterministic vs. machine learning approaches for the analysis phase.
- Section 5 details Phase 4 with respect to IT challenges, including: performance, sustainability, storage, de-duplication, national languages and diacritic characters dependencies, software licensing issues.
- Finally, Section 6 reports the details of each developed pilot, according to a purposefully defined template. It details Phase 3.
2 General Motivations for Web Scraping of Enterprises Web sites

Galia Stateva (BNSI, Bulgaria)

The purpose of this workpackage is to investigate whether web scraping, text mining and inference techniques can be used to collect process and improve general information about enterprises. In particular, the aim is twofold:

1. to demonstrate whether business registers can be improved by using web scraping techniques and by applying model-based approaches in order to predict for each enterprise the values of some key variables;
2. to verify the possibility to produce statistical outputs with more predictive power combined or not with other sources of data (survey or administrative data). The benchmark of big data output could be the data produced by the “ICT use by enterprises” survey. It is carried out in all EU Member States.

Companies tend to be very active in the internet. Their internet activities range from selling products, looking for employees, sharing info about their new products and investments in the future to tweeting. By collecting this information and presenting it in a structured way we can gain an insight what is going on in the economy. This information is much timelier than national accounts and standard business surveys. National accounts in particular may lag economic processes and have no predictive power.

The initial identified use cases were:

1. Enterprise URLs Inventory. This use case is about the generation of a URL inventory of enterprises for the Business register.
2. E-Commerce in Enterprises. This use case is about predicting whether an enterprise provides or not web sales facilities on its website.
3. Job vacancies ads on enterprises’ websites. This use case is about investigating how enterprises use their websites to handle the job ads.
4. Social Media Presence on Enterprises webpages, aimed at providing information on existence of enterprises in social media.
5. Sustainability reporting on enterprises’ websites. One of Sustainability Development Goals target set up by the UN is to encourage enterprises to produce regular sustainability reports highlighting the sustainability actions taken. In order to measure companies’ response to this, this use case will look at what companies publish on their official website and track changes over time.

Use case 3 is particularly useful for WP1 to understand if the enterprises’ websites can be used as information channels for WP1. In general, WP2 faces some challenges not faced in WP1, including application of much larger-scale scraping of websites and collecting and analysing more unstructured data.

Work package 2 has six participating countries, namely IT (leader), BG, NL, PL, SE and UK.
The main activities carried out within the work package are:

- **Task 1 – Data access**
- **Task 2 – Data handling**
- **Task 3 – Testing of Methods and Techniques**

Within Task 1, the main activities were: (i) identification of a set of methods to retrieve URLs for enterprises for which they were not available and (ii) study of the legal aspects to access data on enterprises web sites.

URL identification is the very first step toward getting access to company information and presenting it to decision makers in a structured way.

The study of the legal aspects was a particularly important initial step. Each participating country had to involve internal legal offices and there were much iteration before getting the desired answers. The result of this activity is the Deliverable 2.1 “Legal aspects related to Web scraping of Enterprise Web Sites”.

Within Task 2, the main activities were: (i) detailed definition of use cases and (ii) carrying out scraping activities.

The detailed definition of use cases was carried out according to a shared template by all the participating countries which have been published on the project wiki platform (https://webgate.ec.europa.eu/fpfis/mwikis/essnetbigdata/index.php/WP2_Worlding_area).

In order to carry out the scraping activities, some software tools needed to be developed and shared. All the countries worked on performing scraping activities according to the defined use cases. Some countries, namely SE and UK, are not able to perform massive scraping, though they can work on scraping a limited number of sites.

Within Task 3 the main activities were: (i) selection of some use cases, among the defined ones, that enable a good representativeness of the overall potential statistical outputs and information to enrich business registers; (ii) build of a proof of concept for each of the selected use cases to predict characteristics of the enterprises by applying text and data mining techniques.

The selected use cases for SGA1 are:

1. Enterprise URLs Inventory.
2. E-Commerce in Enterprises.
3. Job vacancies ads on enterprises’ websites.

The remaining 5 and 6 use-cases, mentioned above will be implemented in SGA2.

The distribution of countries participating to each selected use case are shown in Figure 1 (the bold X identifies the country responsible for the use case).
In the pilots implementation, UK and SE are not able to perform massive web scraping due to legal reasons as detailed on the WP 2 deliverable 2.1 “Legal aspects related to Web scraping of enterprise web sites”.

Use case analysis results

As remarked a detailed use case definition was carried out by first sharing a use case template. The use case template involves the following fields:

- ID
- Name
- Description
- Actors
- Preconditions
- Postconditions
- Frequency of use
- Scenarios
- Special Requirements
- Issues

Participant countries filled the template concerning the use cases they are involved in.

All the use cases specified according to the template are available on the project wiki (https://webgate.ec.europa.eu/fpfis/mwikis/essnetbigdata/index.php/WP2_Working_area).

3 Description of a Reference Framework for Web Scraping of Enterprises’ Websites
Monica Scannapieco (Istat), Donato Summa (Istat)

From a conceptual point of view we designed a generic reference logical architecture made of several building blocks organized into four main layers (see Figure 2), namely: « Internet access », « Storage », « Data preparation » and « Analysis ». For
each layer we reported the logical functionalities to be implemented by specific software products; so, for instance, the «Internet access layer» has two logical functionalities, i.e. «URL searcher» and «Scraper».

The objective of the **URL searcher** block is to retrieve a list of websites related to a given enterprise. Usually this list is obtained by querying a search engine on the web using the name of the enterprise as a search term. The underlying assumption is that, if an enterprise has an official website, this should be found within the results provided by a search engine.

The **Retrieved URLs** block is basically a container of URLs obtained in the very previous step, it can be implemented in different ways, ranging from simple file to a DBMS.

The **Scraper** block is responsible for acquiring the content available on each URL in the list of URLs provided as input. It can have additional features such as URL filtering (if a list of URL to filter out is provided) and is usually configurable by setting different parameters such as the level of scraping (e.g. Just the homepage or homepage plus first level etc.).

The **Scraped content** block is a container of the content scraped by the Scraper block. Usually it is necessary to implement this block using non trivial solutions due to the fact that the amount of information could be huge and made of unstructured data.

The **Index configuration** block represents a strategy of indexing the scraped data stored into the Scraped content block. In a Big Data context the huge amount of data that can be stored is a parameter that has to be taken in great consideration, data indexing is usually necessary in order to easily retrieve information in subsequent phases. This block is normally included in the storage platform.
The **Feature extraction** block is responsible for localizing and retrieving from a scraped resource a set of predefined features of interest (e.g.: addresses, telephone numbers, names, VAT codes, etc). Usually it is implemented in a SW program.

The **URL scorer** block is used to assign a score to an URL on the basis of some predefined parameters such as the presence of some features of interest inside the URL’s content. Given a list of URLs related to an enterprise, this block can be used alone or in conjunction with other block in order to identify the most probable official URL for that particular entity.

The **Tokenization** block processes the textual content of the scraped resources by transforming it in a text that becomes input for further processing such as parsing and text mining or for analysys blocks. Normally, in lexical analysis, tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens.

The **Data parsing** block focuses on the analysis of the tokens produced by the Tokenization block by searching for a specific regular expressions, matching sentences etc.

The **Word filters** block is used to filter out some words/tokens (if a list of words to be filtered out is provided) from the scraped textual content or to enrich it with a list of go words.

The **Language specific lemmatization** block lemmatizes the tokens found in the scraped textual content in order to reduce the number of textual elements to be analized. Lemmatization in linguistics is the process of grouping together the inflected forms of a word so they can be analysed as a single item, identified by the word's lemma, or dictionary form. In this case (computational linguistics), lemmatization is the algorithmic process of determining the lemma of a word based on its intended meaning. When it is not possible to desume the intended meaning usually the base form of a token is obtained by using a stemmer that compute the base form of a token by operating on a single word without knowledge of the context.

The **Term document matrix generation** block is responsible for producing a TDM (Term Document matrix) to be used by analyses blocks. Normally each cell of the matrix contains the number of occurrences of a token in an enterprises’ website.

The **Machine learning** block (and its sub-blocks) produces the final output statistics by using one (or more) learner(s).

The **Deterministic rules** block is designed from a set of rules with known characteristics of the sites and data in mind.

The **Information extraction-NLP block** performs analyses by relying on Natural Language Processing approaches.

The reference logical architecture has been adopted by all the developed pilots. As detailed in Section 7, each pilot has been developed has an “instantiation” of this architecture.
4 Methodological Issues and Solutions
Olav ten Bosch (Statistics Netherlands), Giulio Barcaroli (Istat), Monica Scannapieco (Istat), Dick Windmeijer (Statistics Netherlands)

In this chapter we focus on the methodological side of web scraping for statistics. We try to answer questions such as:

- “What can we learn methodologically from the four pilots executed in the field of web scraping of enterprises web sites in six countries?”
- “How easy is it to compare the approaches taken in different circumstances and what are the general underlying principles?”
- “Can we identify some methodological best practices, common problems and solutions from the pilots that were executed?”

We will compare and review the pilots according to three dimensions: (i) specific versus generic, (ii) the use of machine learning or deterministic approaches and (iii) the methods used in the pilots. We will use the pilots executed in SGA1 as a framework of reference to describe these issues.

4.1 Specific vs generic scraping: when/what
The crucial factor when web-scraping is whether you know the structure of the content where the information is to be found or whether you cannot make any assumptions beforehand about the structure of the data. We will refer to the first case as Specific scraping, in the second case as Generic scraping.

We define the two different concepts more specifically:

1. We define **Specific web scraping** as when both the structure and the (type of) content of websites to be scraped are perfectly known. In this case, crawlers just have to replicate the behaviour of a human being visiting the website to collect the information of interest. Typical examples of specific web scraping can be found in the area of price statistics, where most of the items in a web store have the same product listing or product page and scraping software can retrieve specific elements, such as the original price, the sales price, the label, the description, the quantity, colour, material etc. from many web pages for many products.

2. We define **generic web scraping** as when no a priori knowledge on the structure and content is available and the whole website must be scraped and processed in order to infer some information of interest. A clear example is the discovery of web pages of enterprises to derive some general characteristics of the population of enterprises. In this case more general scraping methodologies are needed using scraping and processing software.

Of course there are examples where the object of interest has to be attacked by a mix of specific and generic scraping. An example is the retrieval of job vacancies from a job portal, where the main elements of each job vacancy, such as the title, the domain, the branch and maybe closing date are usually well-structured and specifically scrapable. However, the job description tends to be an unstructured piece of text that has to be interpreted using techniques for generic scraping, such as text mining and machine learning.
So why do we make this distinction? We do this because it makes a difference technically and methodologically. The less we know of the subject to be scraped, the more generic our scraping techniques and methodologies should be. Technologically speaking with specific scraping it is possible beforehand to study the object of interest (website or sites) and design a specific scraper that uses this knowledge to navigate through the site using page structure identifiers such as html id’s, xpaths and css selectors. With generic scraping we usually strip out all html markup and apply text and document mining software on the remaining content. From a methodological viewpoint with specific scraping we have reasonably well defined variables (keep in mind that in scraping there is always a certain degree of uncertainty about your data) to use in our processing, whereas in generic scraping we usually apply (machine) learning methods on the results of the data processing steps.

This idea is depicted graphically below:

![Graphical representation of scraping types]

Although advanced text mining and machine learning methods might be applicable to specific scraping contexts as well, we see in the pilots that these methods are particularly applicable to generic scraping. In the next section we will dive deeper into this subject based on the analysis techniques used in the various pilot studies.

### 4.2 Analysis techniques: machine learning vs deterministic approaches

In the analysis phase we see a distinction between methodological approaches based on (machine) learning techniques and approaches based on deterministic techniques. Before we roughly classify our pilots into either one of these two approaches we give an informal definition:

1. We speak about **machine learning** approaches in scraping for official statistics when algorithms or models are derived from a set of training data which is supposed to be reasonably representative for the problem at hand. The parameters of the model are usually tuned with a validation set before measuring its performance on a so called test set with known characteristics. Finally the model is then applied to other sets of data of which we do not know anything but for which we suppose the model performs well, in order to produce statistics.

2. We speak about **deterministic approaches** in scraping for official statistics when algorithms are designed from a set of rules with known characteristics of the sites and data in mind. Put in a different way, the knowledge of an expert is used to design an algorithm to process and interpret input data from web and other sources into statistical target variables. We call this method deterministic, because the algorithm applied to the same data will always have
the same (deterministic) result, where the result of a machine learning approach heavily depend on the training set being used\(^2\)

In general turning web data into statistics usually takes many steps, and in each step a different approach may be taken. One of the factors that influence this choice is the complexity of the relationship between the input variables or the features derived from the input data and the statistical target variables. If this relationship is fairly straightforward a deterministic approach might be obvious. But if this relationship is complex, unknown or difficult to model in an algorithm, which might well be the case when working with web data, a machine learning approach might be the way to go.

One thing to be noted here is that in machine learning approaches the availability of training data of sufficient quality is essential. This happens to be a challenge in many cases. In some of the pilots this training data is available or can be derived from earlier surveys. This might be true the moment when a machine learning approach is introduced in official statistics to (partly) replace a traditional statistical process, however on the long run, survey-based training data might become a rarity and other means have to be found to (re)train machine learning models. Obviously, deterministic approaches do not have this challenge, but have other pitfalls.

The following table (Table 1) roughly classifies the methods used in the pilots into the use of deterministic (D) and Machine learning (ML), knowing that some of them are actually a mix of both:

<table>
<thead>
<tr>
<th>IT</th>
<th>SE</th>
<th>UK</th>
<th>NL</th>
<th>BG</th>
<th>PL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 URLs retrieval</td>
<td>ML</td>
<td>-</td>
<td>D</td>
<td>ML</td>
<td>ML, D</td>
</tr>
<tr>
<td>2 Ecommerce</td>
<td>ML</td>
<td>-</td>
<td>ML</td>
<td>D</td>
<td>D</td>
</tr>
<tr>
<td>3 Job Advertisements</td>
<td>ML</td>
<td>ML</td>
<td>ML</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4 Social Media</td>
<td>ML</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>D</td>
</tr>
</tbody>
</table>

Table 1: Methods used in pilots

This table shows that both deterministic as well as machine learning approaches are used. In the next section we take a closer look at the methods used in the pilots.

4.3 Review of methods used in pilots

In this section we take a look at methods used within the pilots with our classification in specific and generic scraping and deterministic and machine learning approaches in mind.

It is relevant to give a better detail on the machine learning flow in order to have a better understanding of the work done within ML-based pilots.

\(^2\) One could argue that a machine learning approach, once the model has been trained, is in fact also deterministic. That is true, however, these terms seem to be used widely to indicate the difference, so we will continue to use them in this document.
Under the machine learning approach, the approach taken for turning the text collected by generic scraping into a model is crucial. HTML, and other information contained in tags and images, is not structured, and contain a lot of noise which, if not filtered, would make signals unintelligible. For this reason, text and data mining techniques must be applied in order to (i) structure and standardize data and (ii) detect relevant information.

In most of the experiences carried out in use cases, collected texts were processed through the following steps:

1. normalization: stemming and lemmatization;
2. feature selection: for each dependent variable (e-commerce, online job-application, presence in social media) all normalized terms were processed treated in order to detect the most relevant for the prediction: a number of techniques have been employed, such as the correspondence analysis, regularization techniques (LASSO, Ridge, Elastic Net), importance in Random Forest generation.

Once reduced the number of terms to a manageable number of significant predictors, they were used as input to a number of models to be fitted in given training sets (obtained in different ways: by ad hoc analyzing a number of cases, or by using survey data as in the case of the ICT survey). In general, a training set has been partitioned in a real train set, and in a test set, the latter used to evaluate the model by comparing observed and predicted values.

We will now discuss the four pilots in detail.

The URLs retrieval pilot has usually three steps: 1) getting starting points from a search engine, 2) do some scraping on the urls found, or otherwise extract information about them and 3) determining which of the results belongs to the enterprise. This first step is either performed via a structured API or by scraping the results page of a search engine. In either case the first step is a clear example of specific scraping with deterministic analysis techniques - the search results can be viewed as structured content. The second step is an example of generic scraping as nothing is known from the site to be scraped beforehand, except for the UK pilot, which instead utilised an API to get structured information about website registration. In the analysis phase, all countries used an ML approach to determine the correctness of a found url. BG also applied manual validation of results.

The Ecommerce pilots are mostly examples of generic scraping. Starting from a list of enterprise urls, nothing is known about the structure of the website beforehand. However there are certain common characteristics on the way that web shops advertise there commercial activities. These characteristics have been used by NL to design a simple deterministic approach based on a vocabulary of keywords that are commonly used on the home page of a webshop. The other countries used a ML approach for the the identification of Ecommerce activities, where the ICT survey was mainly used as a training set. In the case of Italy, the best models have been used to predict the values of the target variables for all the units in the population of interest for which it was possible to identify the websites and scrape them.

UK used a bag-of-words Naïve Bayes classifier to predict whether a business is engaged in ecommerce. Methods applied are again text mining and the use of vocabularies. We conclude for now that Ecommerce detection seems to be an area where multiple approaches can be successful.
The **Job advertisements pilot** is an example where a mix of specific and generic scraping is applied. Sweden did some generic scraping on public sector websites. In addition they used data from job portals not retrieved by scraping. In the analysis phase Sweden used machine learning approach. Italy used a generic web scraping followed by a ML approach as in the case of the Ecommerce pilot.

In the **Social media presence pilot**, Italy used a generic web scraping followed by a ML approach. Poland developed a script to detect some of the most common social media buttons on websites of enterprises. As it heavily relies on a known structure that this is an example of specific scraping (although the structure can be anywhere on the website). The output is a list of social media characteristics which has not been processed any further. Future work on social media may go one step deeper, not only detecting social media activity but also the *kind of activity*, and for this kind of use-case the scraping will probably evolve into a mix of specific and generic scraping. After all, Twitter messages are always the same, but nothing can be said beforehand on their content and must be interpreted. The same applies to other social media messages.

The following table (Table 2) gives a quick view of the Machine learning classifiers used by ML-based pilots:

<table>
<thead>
<tr>
<th></th>
<th>IT</th>
<th>SE</th>
<th>UK</th>
<th>NL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 URLs retrieval</td>
<td>Neural Networks</td>
<td>-</td>
<td>Support Vector Machines</td>
<td>Support Vector Machines</td>
</tr>
<tr>
<td></td>
<td>Random Forest</td>
<td>Logistic Model</td>
<td>-</td>
<td>Random Forest</td>
</tr>
<tr>
<td></td>
<td>Logistic</td>
<td>Boosting</td>
<td>Gaussian Naïve Bayes</td>
<td>Decision trees</td>
</tr>
<tr>
<td></td>
<td>Neural Net</td>
<td>Bagging</td>
<td>Multilayer perceptron</td>
<td>Naive Bayes classifier</td>
</tr>
<tr>
<td></td>
<td>Naive Bayes</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

| 2 Ecommerce | Support Vector Machines | - | Naive Bayes classifier | - |
|   | Random Forest Logistic | - | - | - |
|   | Boosting | - | - | - |
|   | Neural Net | - | - | - |
|   | Bagging | - | - | - |
|   | Naive Bayes | - | - | - |

| 3 Job Advertisements | Support Vector Machines | - | Naive Bayes classifier | - |
|   | Random Forest Logistic | - | - | - |
|   | Boosting | - | - | - |
|   | Neural Net | - | - | - |
|   | Bagging | - | - | - |
|   | Naive Bayes | - | - | - |

| 4 Social Media | Support Vector Machines | - | - | - |
|   | Random Forest | - | - | - |
4.4 Potential use of pilots’ output

From the previous sections we conclude that it is indeed useful and feasible to apply web scraping techniques in the field of official statistics to compute some experimental indicators. Although the scraping is not always easy to perform and methodologies should be designed carefully, the indicators that can be computed as outputs of the developed pilots are

- URL Retrieval - Rate(s) of retrieved URLs from an enterprises’ list
- Ecommerce - Rate(s) of enterprises engaged in ecommerce from enterprises websites
- Job advertisements - Rate(s) of enterprises that have job advertisements on their websites
- Social media presence:
  - Rate(s) of enterprises that are present on social media from their websites
  - Percentage of enterprises using Twitter for a specific purpose, i.e.
    - a) Develop the enterprise’s image or market products (e.g. advertising or launching products, etc);
    - b) Obtain or respond to customer opinions, reviews, questions;
    - c) Involve customers in development or innovation of goods or services
    - d) Collaborate with business partners (e.g. suppliers, etc.) or other organisations (e.g. public authorities, non governmental organisations, etc.)
    - e) Recruit employees
    - f) Exchange views, opinions or knowledge within the enterprise

Predicted values can be used for a twofold purpose:

1. at unit level, to enrich the information contained in the register of the population of interest;
2. at population level, to produce estimates.

The quality of data pertaining the unit level can be measured by considering the same indicators produced to evaluate the model fitted in the training set. Under given conditions (if the training set is representative of the whole population), the measure of the accuracy (and also of other indicators: sensitivity and specificity) calculated in this subset can be considered a good estimate of the overall accuracy.

In contrast, the evaluation of the quality of population estimates making use of predicted values is much more complex. A complete evaluation should be based on an estimation of the Mean Square Error, that is the joint consideration of the variance and of the bias affecting the estimates. While it is possible to estimate the variance of the estimator making use of resampling methods (in particular, bootstrap), instead the evaluation of the bias implies to know the true value of the parameter in the population, which is a rare condition. Simulation studies, where artificial population sharing...
distributional characteristics with the real one are generated, can support in evaluating all the components of the Mean Square Error.

For instance, in the use cases related to ecommerce, job advertisement, presence in social media, the proportion of enterprise whose websites are characterized by a positive answer to these variables, can be estimated (in different domains of interest) by survey data (with classical sampling estimators, under a design-based and model-assisted approach) and by using Internet data (with a model-based approach). Selection bias stemming from website identification may be particularly important – it may be that URL identification methods are more likely to work well for websites that conduct e-commerce, for example, which will lead to selection bias.

The two sets of estimates can be compared. In order to decide if their difference in terms of quality is relevant, and, in case, which set is the most accurate, the comparison can make use of resampling techniques and simulation studies.

The work done so far could be extended in multiple ways. In particular, if we consider the different use cases:

1. the ecommerce detection algorithms could be refined to distinguish between different levels of ecommerce maturity (for instance, determined by the presence of only an ordering facility, or also payment and deliver tracking ones).
2. The job advertisement spiders could be trained to additionally take the job details and the enterprise characteristics into consideration. The identification of the characteristics of each single job (economic activity, profession) and even the skills required, is a much more ambitious task that implies a different approach, more oriented to “information retrieval” than to “machine learning”.
3. The social media presence detection could be extended to not only scrape the enterprise website, but also the social media itself and its users in order to investigate what kind of use of social media is being done in a more detailed way.

This would all lead to more detailed, but still experimental, indicators.

### 4.5 Conclusions

In this section we look back to the questions raised in the introduction of this chapter and briefly present the most relevant answers.

- “What can we learn methodologically from the four pilots executed in the field of web scraping of enterprises web sites in six countries?”
  One thing we learned is that it is useful and feasible to apply web scraping techniques in the field of official statistics to compute experimental indicators. However, there is not one preferred way of doing these very different pilots in different countries. Even per pilot the methods being used differ, which may have been caused by different data landscapes per country or other circumstantial differences. However some common machine learning methods have been applied in some of the pilots, especially the URL finding pilot where Italy, Bulgaria and the Netherlands applied the same methodology.
- “How easy is it to compare the approaches taken in different circumstances and what are the general underlying principles?”
Using the same terminology and describing the work being done in general building blocks has been a big advantage to compare the different approaches, not only from an IT viewpoint, but from a methodological viewpoint as well. The concept of generic versus specific scraping and deterministic versus machine learning approaches (and within machine learning a wide spectrum of different classifiers) form a useful general underlying basis for scraping for official statistics.

- “Can we identify some methodological best practices, common problems and solutions from the pilots that were executed?”

All of the approaches used in the pilots in the different countries resulted in some draft results which were described in the previous sections. It is very difficult to generalize the approaches being taken. Both deterministic as well as machine learning approaches have been successfully applied. In the latter case one should put energy in finding a training set which might come from survey data (as long as it is available).

5 Technological Issues and Solutions

Jacek Maślankowski (GUS)

Web scraping is not a new method - it has been used since the beginning of the Internet and has evolved in recent years. For instance, proxy servers scraped the content of websites and serve them for local computers more than 20 years ago. This long history of web scraping has resulted in the variety of different tools and methods that can be used to gather information from websites.

Web scraping tools can be divided into generic and dedicated. Typical generic web scraping tools include the following: import.io, Scrapy, imacros, Apache Nutch or similar. The second group includes libraries for a specific purposes, such as Tweepy for Python programmers – to scrap the data from Twitter. The extended list of web scraping tools and libraries can be found on various repositories, including a Github³.

The aim of this section is to give a general overview of the programming languages, tools and libraries used in WP2 pilots. Section 5.1 shows the current software used for pilots implementation. In section 5.2 there is information on the technological issues we have to tackle with during implementation of use cases.

5.1 Review of the technological environments used in pilots

Due to the variety of tools and methods for web scraping, each pilot can be implemented in various way. As mentioned in the previous point, there are repositories with hundreds of different tools and libraries that support web scraping. The goal for the WP2 members was to select popular open source or free web scraping software, that NSI’s employees are familiar with. It was the reason to include traditional as well as big data dedicated software that can be downloaded and implemented in NSI without extra costs.

In Table 3 we included the current programming languages, libraries and tools used in pilots.

³ https://github.com/lorien/awesome-web-scraping, accessed 30th May 2017
<table>
<thead>
<tr>
<th>Use Case</th>
<th>BG</th>
<th>IT</th>
<th>NL</th>
<th>PL</th>
<th>SE</th>
<th>UK</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL Retrieval</td>
<td>(1) PHP language Java URL searcher Jabse Search API Google Custom Search API Bing Search API (2) ISTAT URL Retrieval Java language</td>
<td>Java language R language</td>
<td>Python language JavaScript language NodeJS ElasticSearch engine Natural library for NodeJS Solr/learn library for Python</td>
<td>Java language (ISTAT URLSearcher) Python language</td>
<td>Python language Py-whois API Bing API</td>
<td></td>
</tr>
<tr>
<td>E-commerce</td>
<td>PHP language</td>
<td>Java language</td>
<td>R language</td>
<td>TreeTagger library for lemmatization SnowballStemmer for stemming Crawler4J</td>
<td>Python language</td>
<td>R language</td>
</tr>
<tr>
<td>Job vacancies</td>
<td>Java language</td>
<td>Java language</td>
<td>R language</td>
<td>TreeTagger library for lemmatization SnowballStemmer for stemming Crawler4J</td>
<td>Python language</td>
<td>Libraries: urlib3, urllib, BeautifulSoup, skileam, tensorflow, pandas</td>
</tr>
<tr>
<td>Social media</td>
<td>PHP language</td>
<td>Java language</td>
<td>R language</td>
<td>TreeTagger library for lemmatization SnowballStemmer for stemming Crawler4J</td>
<td>Python language</td>
<td>Sci-kit learn library BeautifulSoup library Tweepy library Apache Spark/Hadoop to execute scripts</td>
</tr>
</tbody>
</table>

**Table 3: Overview of programming languages, libraries and tools used in pilots**

Based on the Table 3, we can say that four different programming languages are used: Python, Java, PHP and R. For Netherlands, Sweden, Poland and United Kingdom the main programming language is Python. For Italy most of the work was done in Java and R. Bulgarian statisticians prefer to use open source PHP programming language.

There are several libraries used for pilots. Regarding Python examples, Poland and Sweden decided to develop a solution based on BeautifulSoup4⁴, while Netherlands and UK decided to use Scrapy⁵ for these purposes. Based on their experience and lesson learnt we can say that Scrapy is a useful, scalable tool for web-scraping. On the other hand BeautifulSoup4 is a HTML parser that can process various forms of websites. For WP2 purposes there is a need to have any HTML parser that is able extract HTML tags from the file and exclude any CSS or JavaScript code in the analysis. It is also suggested to choose tools and libraries that are able to scrap the data from datafiles such as XML/JSON/CSV/TXT. Before scraping the webpage, it is recommended to inspect it by using an inspect tool, such as page inspector⁶ for Firefox web browser. It allows to identify which HTML tags are responsible on the web page to select a specific library.

For text processing and machine learning purposes in Python we have used NLTK⁷ library and scikit-learn⁸. The NLTK library in Python is useful for investigating more advanced natural language processing. In text processing, the use of regular expressions is particularly powerful - especially in

---

⁴ [https://pypi.python.org/pypi/beautifulsoup4](https://pypi.python.org/pypi/beautifulsoup4), accessed 30<sup>th</sup> May 2017
⁵ [https://scrapy.org](https://scrapy.org), accessed 30<sup>th</sup> May 2017
⁷ [http://www.nltk.org](http://www.nltk.org), accessed 30<sup>th</sup> May 2017
⁸ [http://scikit-learn.org](http://scikit-learn.org), accessed 30<sup>th</sup> May 2017
Polish Social Media Presence pilot, it allowed to identify all hyperlinks, even they are not included in the anchor HTML tag. Sci-kit learn allows access to various machine learning algorithms, including Naïve Bayes, SVM, Decision trees or Random forest. R provides a substitute for Python libraries - ISTAT developed several scripts regarding machine learning in R, as presented in Table 3.

The purpose of the choice of the tools for storage layer was to have an environment easy to maintain, and CSV files where therefore a common option. The comparison of the storage use for different pilots was included in Table 4.
<table>
<thead>
<tr>
<th>Use Case</th>
<th>BG</th>
<th>IT</th>
<th>NL</th>
<th>PL</th>
<th>SE</th>
<th>UK</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL Retrieval</td>
<td>MySQL</td>
<td>Apache Solr</td>
<td>CSV</td>
<td>CSV</td>
<td>CSV</td>
<td>CSV</td>
</tr>
<tr>
<td></td>
<td>(2) ISTAT software: Apache Solr</td>
<td>Apache Solr</td>
<td>CSV</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E-commerce</td>
<td>MySQL database</td>
<td>Apache Solr</td>
<td>CSV</td>
<td></td>
<td></td>
<td>CSV</td>
</tr>
<tr>
<td>Job vacancies</td>
<td>MySQL database</td>
<td>Apache Solr</td>
<td>CSV</td>
<td>CSV</td>
<td>CSV</td>
<td>CSV</td>
</tr>
<tr>
<td>Social media</td>
<td>MySQL database</td>
<td>Apache Solr</td>
<td>CSV</td>
<td>CSV</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 4: Data storage used in pilots**

The most common way of data storing is a filesystem with CSV file type. Selected pilots use the Apache Solr database (NoSQL) and MySQL database (relational). The decision of using the filesystem as a primary data storage is a result of the fact that most of the tools used in pilots have embedded libraries to access CSV files. On the other hand, CSV files are used to store the results of analysis. The use of this filetype helps to load it into popular applications, such as R or MS Excel. To increase the performance of accessing such files, there is a possibility to store them in HDFS filesystem (Hadoop Distributed File System) to perform automatic and very efficient parallel data access.

The purpose of the use of Apache Solr is to provide a scalable environment that is able to store different types of data. However the main purpose of the use of Apache Solr in WP2 pilots is to store websites in NoSQL database. This type of database allows dynamic searching through its storage including full-text search, hit highlighting, faceted search, dynamic clustering, database integration, rich document handling, distributed search, index replication and high scalability.

For the WP2 purposes several different tools has been developed. For example, ISTAT developed a software URLSearcher in Java that allows to retrieve URLs associated with different enterprises based on their attributes, such as name, city, contact address. This tool can be applied in Java environment in every operating system. The output of this tool is a CSV file or a set of files including links identified. Further analysis of URLs can be performed with other tools developed by ISTAT for the URL Retrieval use case. The list of tools used or developed for pilots was included in Table 5.
<table>
<thead>
<tr>
<th>Use Case</th>
<th>BG</th>
<th>IT</th>
<th>NL</th>
<th>PL</th>
<th>SE</th>
<th>UK</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL Retrieval</td>
<td>URLSearcher (custom Java application) RootJuice (custom Java application) UrlScorer (custom Java application)</td>
<td>URLSearcher (custom Java application) RootJuice (custom Java application) UrlScorer (custom Java application)</td>
<td>URLSearcher (custom Java application) RootJuice (custom Java application) UrlScorer (custom Java application)</td>
<td>Python language</td>
<td>Custom Python scripts, including scrapy applications and utilising NLTK Library (Naïve Bayes)</td>
<td></td>
</tr>
</tbody>
</table>

**Table 5: Tools used or developed for the pilots**

Most of the software was developed as a short scripts used for a specific actions regarding data collection and processing. The most popular environment for the pilots testing was to use virtual machine (e.g., Linux Mint) or dedicated machine (e.g., Linux Ubuntu Server, MS Windows).

Our experience of pilots sharing shows that they WP2 scripts and applications are platform independent. For instance, it is possible to run Python scripts or Java applications in Linux as well as Windows environment without any changes in the source code.

During our pilots implementation we have learnt that Apache Solr that is not only a NoSQL database but also an enterprise search platform. It is possible to store any type of data, including web pages, which represents the main input files of the WP2 cases.
5.2 Issues and Solutions

Performance
The performance is a key issue especially when downloading and processing the whole websites. Processing unstructured information is very CPU and memory consuming, especially with machine learning algorithms, and as a result not very efficient. Because of the fact that most of the environment used for WP2 pilots have limited CPU and memory resources it was difficult to estimate how efficient the algorithm will be in a production environment. Based on the Bulgarian example we can say that conventional IT tools are sufficient for the URL inventory creation with tens of thousands enterprises. Based on the Italian experience, Apache Solr has technical problems in dealing with Solr Connection pool in the loading phase right after the scraping phase.

Sustainability
Due to the fact that big data tools are changing very frequently as well as the website technology, there is a need to provide an agile-like development of tools. The technologies used for the pilots are likely to change in the next few years, and for that reason we do not recommend a specific language. One example of this is the Polish implementation of the social media use case, which was started in Python 2 with HTML Parser as the main library and was eventually migrated to Python 3 with BeautifulSoup library.

It is possible to switch easily to other platforms. ISTAT example allows to use another similar solution, e.g., elastic Search instead of Apache Solr. The use of CSV files implemented in most of the use cases can also be replaced by any filesystem more efficient, such as HDFS.

Storage
Based on the Bulgarian experience, the requirements for the storage of a database about 27,000 enterprises takes around 1 GB of HDD, including BR data, scraped Search APIs data and enterprises web sites and e-stores first pages titles, key words, descriptions and URLs data. It allows to create a conclusion that for this specific use case it is possible to use traditional technology.

As mentioned in the previous section of this document, the possible choices are between filesystem (CSV, JSON etc.), NoSQL database (Solr, Cassandra, Hbase etc.) or relational database (MySQL, PostgreSQL, SQL Server etc.). The decision of the use of particular data storage should be taken according to the volume of the data and the type of data to be stored.

De-duplication issues
There is a need for de-duplication framework that will automatically exclude all duplicates of websites and particular information taken from them, e.g., job offers. Based on the Swedish experience, de-duplicating the pages is needed since the same pages are retrieved from different links.

National languages and diacritic characters
Although the frameworks are developed in particular countries, it is possible to apply them in other country without any major changes. For example, URLSearcher, a tool developed by ISTAT was tested with Polish websites that contains specific diacritic characters, such as “ą”, “ę” or “ś”. Since the results are stored in UTF-8 coding page (65001), it allows to be read by other tools, e.g., MS Excel or MS SQL Server via import/export wizard.
However the web scraper must be able to recognize different coding pages when scraping the web page. Polish web pages are usually published in UTF-8, WINDOWS-1250 (CP-1252) or ISO-8859-2. Therefore when storing the data from web pages, it is recommended to unified the coding page. Because of possible international comparability the suggested option is to use UTF-8.

**Licensing**

All software used for the WP2 pilots implementation is free and open-source. This means that everyone can easily test and improve the tools. On the other hand, using such software means and not always we could rely on a good documentation or on a detailed guide to make everything work.

### 5.3. Lessons Learnt and Conclusions

A variety of different tools and methods used in pilots may lead to the question whether it is possible to achieve a high quality output using different tools. The simple answer is yes, as programming languages and data storage technologies are independent from the methodology. It means that we can choose what software is the most convenient for us to solve a specific problem, such as collecting a particular information from unstructured data published on websites.

In fact the decision of using a specific programming language may be dependent on the skills of IT staff employed in NSI. This is the reason why there is such a long list of them, presented in Table 3 of section 5. Different criteria are used when selecting types of data storages to be used. The reason for the use of a specific storage is based on the type of the data that will be stored. When it is necessary to store a whole webpage, the most expected is a database with NoSQL features, such as Apache Solr (see ISTAT Use Case 1). When there is a necessity of gathering a structured data, any structured or quasi structured database or file can be used (such as MySQL or CSV file).

However results of analysis may vary depending on the methodology used. It means that the tool prepared with the specific Machine Learning algorithm, e.g., Naïve Bayes, may provide different results, depending on the training dataset used or the type of an algorithm. It was briefly explained in the appendix by comparing results of the use of different machine learning algorithms (see Appendix, Example 1 IT 1, Table 6).

To conclude, the lesson learnt from implementation of the surveys led to the following conclusions:

- results of analysis are independent on the platform or programming languages used,
- data storage must be appropriate to achieve the goal of the specific use case, the selection criteria can be performance, possibility of data processing and analysis, language used (SQL vs. NoSQL) etc.,
- specific libraries for different programming languages, such as sci-kit learn for Python, have options for algorithm “tuning”; it means that different tool may produce different results even the same library and training dataset is used.
6 Future Challenges in Web-scraping Enterprise Websites
Matthew Greenaway (ONS), Ingegerd Jansson (SCB), Dan Wu (SCB)

This section summarises the key future challenges for web-scraping enterprise websites.

Bias

Some of the pilots have highlighted the issue of bias – for example, we may be more likely to identify websites for businesses which conduct e-commerce, and e-commerce statistics based purely on data scraped from the websites we’ve found will therefore be biased without adjustment. We’ve also found that some websites – for example, those that make heavy use of Javascript – are harder to scrape than others, and this may also introduce bias. A simpler source of bias, for some statistics, will simply be that some businesses are less likely to have a website than others.

A key challenge will be to understand these biases for different use-cases and work out how to adjust for them so that scraped data can be used for estimates qualifying as official statistics. This is likely to involve methodological work around combining web-scraped data and survey or administrative data, potentially picking up on some of the methods that are being developed as a part of WP1.

Ethics

More and more data pertaining to individuals and enterprises is being placed online. This is a significant opportunity, but also a challenge in that it is likely to lead to more concern amongst the public about how government is utilising online data, including relatively uncontroversial cases such as NSIs utilising textual data on company websites. As a response to this, NSIs and Eurostat will need to work to develop transparent web-scraping policies in order to allay public concerns about the data they are collecting and how they are using it. The ‘netiquette’ developed as a part of this work-package is an important first step towards this (see Deliverable 2.1).

The evolving internet

The web-scraping methods utilised in this Work Package are entirely based on collection of textual information from web-pages. This may become increasingly challenging as the internet evolves, as more and more data may be encoded in forms that are harder to extract – audio or video files, for example, or increasing use of interactive or user-specific content. It is therefore easy to imagine that websites for some enterprises – potentially in particular larger enterprises, or those in the creative industries - may become much more challenging to extract information from.

Such web-scraping may therefore need to be increasingly carried out by specialists inside or outside of NSIs, rather than only data scientists.

Data management at NSIs

One challenge which work-package participants had to address was how to carry out web-scraping from on-network machines, or how to transfer scraped data from off-network to on-network storage. Another challenge is engineering the complex data scraped at large scale before further analysis, e.g. applying machine learning to validate, link and integrate data.
We need data warehouse similar systems that can manage the entire data life circle i.e., the storage of data scraped, the stage area for engineering the massive data, and databases for maintaining the clean data.
7 Appendixes by Use Cases

7.1 Use Case 1 - URL Inventory

Common Approach

Five countries carried out pilots for this use case:

- Bulgaria (two pilots: one using own software, one using Italian software)
- Italy
- Netherlands
- UK
- Poland (utilising Italian software)

The approach of all countries followed the same basic outline:

1. Creating a training set of enterprises with matched URLs. Only enterprises with >10 employment were included.
2. Utilising a web-search API to search for either the enterprise name, or the enterprise name followed by ‘contact’, and storing the first 10 results as ‘candidate’ websites
3. For each candidate website, utilise web data to get details. This may be scraped data from the company website, the snippet from the search API result, or a ‘whois’ lookup.
4. Using the collected data to identify websites, either using an algorithm or manually (Poland did not carry out this step)

Details of approach and differences between countries

1. Creating a training set
   Different countries have different availability of website data for enterprises, so different approaches were used. Italy utilised 73,000 URLs, taken from both their ICT survey and third-party data. Netherlands, Bulgaria and Poland all utilised website data on their business register (Netherlands: 1,000 businesses, Bulgaria: 27,000 businesses). UK utilised manually-identified websites for 300 businesses. In all cases, businesses with 10+ employment only were included, although Netherlands included businesses with 0-10 employment in a separate pilot.

2. Utilising a search API
   UK, Italy, Poland and Bulgaria (in their pilot using Italian software) used the Bing search API, while Netherlands used the Google search API. Bulgaria, in their pilot using their own software, used a number of search APIs – Jbase (a Bulgarian search engine), Google and Bing - with the aim of evaluating differences between the APIs and language differences. All countries queried the API with the enterprise name, except Netherlands, who searched for the enterprise name followed by the string ‘contact’ in an attempt to maximize the chance of
an address being present in the response. Poland also trialled using the company name and city name.

3. Data collected

Italy, Poland, the Netherlands and Bulgaria (when testing the Italian software) all used data scraped from the candidate websites. Netherlands also used ‘snippets’ returned by the search API, and relied more heavily on this data. The UK used website registrant information provided by a ‘whois’ lookup – all websites must provide name and address details to either national or global registrars.

4. Website identification

Italy and the Netherlands created, for each candidate website, features based on whether enterprise details could be found in the collected data. The features used by Italy were based on the presence of the enterprise’s telephone number, VAT code, and geographic details on the candidate website. Netherlands used similar features. Italy fit a logistic regression model with these features as independent variables, and accept/reject a website based on the predicted probabilities from this model. They chose a threshold predicted probability of 0.7, above which a candidate website is identified as a genuine match, resulting in recall of 66% and precision of 88%. Netherlands used a different supervised machine learning algorithm.

In contrast, Bulgaria, in their pilot using their own software, used experts to manually identify businesses using the data returned by their API. The UK carried out a simple exact match between the enterprise postcode and the registrant postcode. Poland were simply concerned with evaluating the URL-searcher software, and did not carry out the website identification step.

Summary of findings

- Most countries were able to identify a large number of enterprise websites using the basic methodology of querying a search API with the enterprise and then matching web data to enterprise details. For example, Italy’s pilot identified 95,000 out of an estimated 130,000 URLs before any clerical intervention.

- Most countries identified both false positives (websites incorrectly identified) and false negatives (websites not identified), and some countries identified ‘borderline’ requiring clerical input. Some clerical intervention will always be required in order to build a URL inventory with good accuracy.

- A variety of web data may be useful in choosing between candidate websites from search API – including data scraped directly from the websites, ‘snippets’ from search results, the ranking of search results, and website registration information. However, it is insufficient to use registration information only, meaning some data must be collected from websites. This means all countries interested in website identification must address the legal and ethical issues around web-scraping.
• Where a website identification method cannot identify all enterprise websites, it is important to consider bias stemming the probability of any given website being identified. The UK pilot found that websites for businesses which conduct e-commerce were notably more likely to be found, which could easily cause bias in estimates.

• Bulgaria and Poland were successful in applying the Italian URL-searching software to their own business data. This suggests that common approaches, and software, may be used in different countries to deal with identifying websites.

7.2 List of Pilots: Use Case 1

Pilot identification

“1 BG 1” (first pilot implemented by Bulgaria regarding use case number 1)

Reference Use case

<table>
<thead>
<tr>
<th></th>
<th>1) URL Inventory of enterprises</th>
<th>2) E-commerce from enterprises’ websites</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3) Job advertisements on enterprises’ websites</td>
<td>4) Social media presence on enterprises’ web pages</td>
</tr>
</tbody>
</table>

Synthesis of pilot objectives

The BNSI used the business register information for enterprises which form the ICT survey population for URLs inventory. The main objective was generated a URLs inventory of enterprises. The Inventory has been used for web-crawling of the enterprise sites to retrieve information for e-commerce and social media activities. Approaches that were used are the following:

• Use the Jabse (Just Another Bulgarian Search Engine) Search API, Google Custom Search API and Bing Search API (from Pilot 1 BG 2) on the base of the enterprise’s name with filtering of the Search APIs results;

• Retrieve URLs from data sets that contain organization number, URLs, contact details and other enterprise characteristics from the Business register;

The population subject to web crawling is enterprises with 10 or more employees: company name, e-mails, URL and other characteristics from BR. The total size of population was 26 836 companies, 20649 e-mails and 2006 URLs.
**General description of the flow on the logical architecture**

The URL Searcher uses information for URLs and e-mails from BR to check, verify and generate domain names in order to retrieve URLs Inventory. The Scraper uses the enterprise names employing Jabse Search API, Google Custom Search API and Bing Search API to get sets of up to 10 suggested URLs. The Searcher and Scraper store the information in the DB. The true enterprise URLs are identified by DB crawling interface used by experts for manually validation. In the analysis phase the statistical results were calculated with specific software script.

**Functional description of each block**

The URL Searcher obtains the list of enterprises with available URLs and e-mails from Business register (total number of enterprises: 26836, 2006 initially available URLs, 20649 available e-mails). Then the URL Searcher checks if the initial URLs are real websites and saved the results in the database. If the URLs are not verified or missing, then the URLs are generated from the domain names of e-mails by excluding popular e-mail services (like gmail, yahoo, etc.), if they are available. The generated URLs are verified for existence by the Searcher. All verified URLs are stored within DB (7038 URLs).

The Scraper uses the automated search interface of Jabse, gets up to 10 search results for the businesses from its names in Bulgarian and gets up to 10 search results for the businesses from their names transliterated in Latin. Then the Scraper excludes from the search results the complex URLs, gets just those up to domain names, and suggests them as most probable.

The results are saved in the database in text and html format (15638 sets of up to 10 most probable search results in Bulgarian, 16201 sets of up to 10 most probable search results in Latin).

After that, the Scraper uses Google search interface, gets up to 10 search results for the businesses from its names in Bulgarian and saves the results in the database in json format (26829 sets of up to 10 search results).
The DB crawling interface was used by the experts to choose the real URLs of the enterprises from the suggested URLs from the URL Searcher and Scraper. The results of this phase are 9809 real URLs of businesses.

The results and statistics script gives the real-time information for enterprise URLs.

**Description of the technological choices**

BNSI did not have any particular experience with web scrapping and Big Data before this project. So, the first choice of tools for this project were free software web tools that the BNSI has some experience with – Apache web server, MySQL database and PHP programming language. We used PHP for the projects’ software, we used MySQL for the storage platform and we used Apache for execution of PHP scripts over the web browsers. We integrated and used the Jabse Search API, Google Custom Search API and Bing Search API (form Pilot 1 BG 2) in our software to get results suggested by these search engines. The PHP scripts were executed in browser with use of HTML content refresh meta tag (for example: the script queries the Search APIs every 3 seconds with enterprise data and stores the information gathered in the database).

**Concluding remarks**

**Lessons learned**

- Methodology: The Google Search API gives the best results. It gives 200 searches per day free of charge or 1000 for 5 EUR to max of 10000 searches per day. The Jabse database do not cover all the enterprises. Jabse Search works better with its English version then the Bulgarian, but the Search API covers only the Bulgarian version. Overall, the 26836 records were manually checked by the experts in 45 working days, which gives around 600 records per workday.
- IT: Conventional IT tools are sufficient for the URL inventory creation with tens of thousands enterprises. The size of the database about 27000 enterprises takes around 1 GB of HDD (BR data, scraped Search APIs data and enterprises web sites and e-stores first pages titles, key words, descriptions and URLs data).
- Legal: There were no legal issues, because we used third party information tools (Jabse, Google and Bing Search APIs) to obtain the URLs of the enterprises.

**Open issues**

There are no open issues in this pilot. All defined activities in the use-case were successfully executed.
Pilot identification

“1 BG 2” (second pilot implemented by Bulgaria regarding use case number 1)

Reference Use case

[X] 1) URL Inventory of enterprises
[ ] 2) E-commerce from enterprises’ websites

[ ] 3) Job advertisements on enterprises’ websites
[ ] 4) Social media presence on enterprises’ web pages

Synthesis of pilot objectives

The BNSI used the business register information for enterprises which form the ICT survey population for URLs inventory with automated URL retrieval procedure (applying ISTAT software). The main objective was generated a URLs inventory of enterprises. The Inventory has been used for web-crawiling of the enterprise sites to retrieve information for e-commerce and social media activities. The BNSI decided to test the ISTAT URLs Inventory software how it is working on the Bulgarian BR population of enterprises. This has been done to achieve the one of the project’s aim – to share good experience and best practice among the involved countries. In addition, the results from this pilot are going to be used to benchmark with the pilot 1 BG 1 results. Then we intend to evaluate the results from the two pilots.

Initial data are enterprises with 10 or more employees, with their names, contact e-mails, web sites URLs and other characteristics from BR. The total size of population was 26836 businesses, with 20649 e-mails and 2006 initial enterprise URLs.

Pilot details
**General description of the flow on the logical architecture**

The URLSearcher uses the enterprise names with Bing Search API to be able to get sets of up to 10 suggested URLs and saves the obtained information in txt file. The RootJuice takes the txt file, scraps the content of the enterprise web-sites and saves the information in csv file. The csv file information is uploaded in the Apache Solr open source enterprise search platform.

**Functional description of each block**

The URLSearcher obtains 2 files containing the list of the firm names and the corresponding list of firm IDs from Business register (total number of enterprises: 26836). For each enterprise the program queries the Bing Search engine 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 that is a txt file format with all results.

The RootJuice program takes as input 3 files:
1) The seed file from URLSearcher
2) A list of url domains to avoid (usually directories domains, yellow pages and etc.)
3) A configuration file

The RootJuice program tries to acquire the HTML page for each row of the seed file (if the url is not in the list of the domains to avoid) From each acquired HTML page the program selects just the textual content of the fields we are interested in and write a line in a CSV file.

The CSV file from RootJuice is imported in open source storage platform Apache Solr version 6.5.0

**Description of the technological choices**

BNSI uses suggested open source software from ISTAT: Java Run time environment for URLSearcher and RootJuice programs and Apache Solr Storage platform.

**Concluding remarks**

**Lessons learned**

- Methodology: No methodology lessons learned at this stage.
- IT: Since the suggested Apache Solr from ISTAT was different from the version 6.5.0 used in the BNSI, we copied the types definitions from ISTAT schema.xml file of the provided solr installation to managed-schema file from BNSI solr 6.5.0 installation in order to be able to import the csv results in the storage platform.
- Legal: There were no legal issues, because we used third party information tools (Bing Search APIs) to obtain the URLs of the enterprises.

**Open issues**

The next steps of the ISTAT URL retrieval flow (URLScorer and URLMatchTableGenerator) are going to be performed as soon as they will be available on GitHub and customize to be workable in the Bulgarian context.
**Pilot identification**

1 IT 1 (first pilot implemented by Italy regarding use case number 1)

**Reference Use case**

- X 1) URL Inventory of enterprises
- 2) E-commerce from enterprises’ websites
- 3) Job advertisements on enterprises’ websites
- 4) Social media presence on enterprises’ web pages

**Synthesis of pilot objectives**

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 can be searched using a search engine and the obtained URLs must be scraped, stored and analyzed. In the analysis phase the list of enterprises with a website known in advance will be used as a training set for the learners, while the remaining enterprises and their associated URLs found by the procedure will be used as a test set.

**Pilot details**
General description of the flow on the logical architecture

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

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

Once we have the scraped information stored in Solr as documents (one Solr document per URL) we launch 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 is known (training set), we use 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 apply the model to the subset of enterprises for which the URL is not known, in order to decide if an automatically found URL is acceptable or not.

Functional description of each block

UrlSearcher 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 obtaining 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 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 know if the link is correct or not, a dichotomous variable `correct_Yes_No` says if the URL is 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 is 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 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). Their values are reported in Table 6.

<table>
<thead>
<tr>
<th>Learner</th>
<th>Accuracy</th>
<th>Sensitivity</th>
<th>Specificity</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural Networks</td>
<td>0.7960</td>
<td>0.8011</td>
<td>0.7890</td>
<td>0.8194</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.7999</td>
<td>0.8278</td>
<td>0.7616</td>
<td>0.8270</td>
</tr>
<tr>
<td>Logistic Model</td>
<td>0.7918</td>
<td>0.7857</td>
<td>0.8002</td>
<td>0.8135</td>
</tr>
</tbody>
</table>

**Table 6: Evaluation of Neural Networks, Random Forest and Logistic Model.**

The difference in performance is not significantly different for the three learners, this can be seen also visualizing the ROC and the curves of precision/recall in Figure 3.
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. In Figure 4, the fitting results of the logistic model applied to the training set are shown.

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 (Table 7).

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 acceptance threshold value is set to 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.
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.

For our case study, i.e. the survey “ICT usage in enterprises”, the population of interest of the survey is composed by enterprises with at least 10 employees and operating in different branches of industry and services, the size of such a population is around 200000. By the ICT survey estimates, it is known that about 70% of these enterprises do own a website, used for different purposes.

Starting from our input train set (specifically the part of it with at least one page fetched, of size 73003) a logistic model has been fitted, and threshold values chosen, to be used in order to find additional URLs for remaining enterprise websites.
So, on the complementary set of enterprises for which the URLs are not available, the three steps procedure (searching, crawling and scoring) has been applied. Here, we report the results of the application of the logistic model to the 106019 enterprises for which URLs were not available (i.e. 205759-73006=132753, of these at least one link has been crawled for 106019):

- 26097 (24.6%) URLs have been evaluated as reliable, and associated to the corresponding enterprises;
- 68885 (64.9%) have been considered as erroneous, and excluded;
- 11037 (10.4%) have been addressed to interactive controls.

This latter component is expected to contain about 45% of false positive (see Table 2, the 5th class). So, after the controls about 55% of the 11037 URLs, let us say 6000, should be individuated as correct.

In conclusion, at the end of the process we should obtain a total number of identified URLs equal to about 105000. If we consider a total amount of 140,000 websites pertaining to the enterprises population (70% of reference population), we obtain a coverage of near 75%, which can be deemed a satisfactory one.

**Description of the technological choices**

- We developed a set of ad-hoc Java programs, including: URLSearcher, RootJuice and URLScorer.
- We used Bing search engine because it let us execute a great number of automatic queries for free without any restriction.
- All of the programming was done in Java and R due to in-house expertise.
- Due to the particular domain (Big Data) we decided to use Solr that is not only a NoSQL DB but also an enterprise search platform usable for searching any type of data (in this context it was used to search web pages). In fact its major features include full-text search, hit highlighting, faceted search, dynamic clustering, database integration, rich document handling, distributed search, index replication and high scalability.
- In order to decouple the logical layers and because it is a very common and easy to manage data format, we often used csv files to store intermediate data.
- When it was possible we wrapped up already existing pieces of SW (e.g. Crawler4J)

**Concluding remarks**

**Lessons learned**

- Methodology: a machine learning task was set up for this use case. The used learners (namely Neural Networks, Random Forest and Logistic Model) proved to have similar performance. Taking into account the statistical properties of the logistic model, and in particular because of the interpretation of the score as a probability this learner has been preferred to the others.
- Legal: there was an issue related to the massive search engine queries to be done according to the Terms of Use conditions provided by the different search engines (Bing, Google, etc.)
- IT: all used SW is free and open-source, this means that everyone can easily test and improve the tools and the whole process for free; this implies an initial effort that can be difficult to estimate because not always you can rely on a good documentation or on a detailed guide to make everything work.

In terms of performance of the used IT tools, some details are provided in the following.
We set up a testing environment with the characteristics shown in Table 8. Let us notice that, being the task massive and resource consuming, this environment is quite under-sized (in terms of RAM and CPU).

<table>
<thead>
<tr>
<th>CPU</th>
<th>2 cores running at 2.2 GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM</td>
<td>16 GB</td>
</tr>
<tr>
<td>O.S.</td>
<td>Red Hat Enterprise Linux 7</td>
</tr>
<tr>
<td>Kernel version</td>
<td>3.10.0-327.22.2.el7.x86_64</td>
</tr>
</tbody>
</table>

**Table 8: Environment configuration**

<table>
<thead>
<tr>
<th></th>
<th>TrainSet</th>
<th>TestSet</th>
</tr>
</thead>
<tbody>
<tr>
<td># of enterprises</td>
<td>81912</td>
<td>132753</td>
</tr>
<tr>
<td>UrlSearcher execution time</td>
<td>14h 3min</td>
<td>22h 17min</td>
</tr>
<tr>
<td># urls in seed file</td>
<td>814577</td>
<td>1321323</td>
</tr>
<tr>
<td>UrlCrawler execution time</td>
<td>8h 39min</td>
<td>13h 4min</td>
</tr>
<tr>
<td># urls filtered out</td>
<td>470039</td>
<td>846052</td>
</tr>
<tr>
<td># urls after filter</td>
<td>344538</td>
<td>475271</td>
</tr>
<tr>
<td># urls reached</td>
<td>241202</td>
<td>305488</td>
</tr>
<tr>
<td>% of reached urls</td>
<td>70.01</td>
<td>64.27</td>
</tr>
<tr>
<td># of enterprises found</td>
<td>76976</td>
<td>117998</td>
</tr>
<tr>
<td># of enterprises with 0 pages fetched</td>
<td>3970</td>
<td>11979</td>
</tr>
<tr>
<td># of enterprises with at least 1 page fetched</td>
<td>73006</td>
<td>106019</td>
</tr>
<tr>
<td>Output CSV file size</td>
<td>8.6 GB</td>
<td>10.1 GB</td>
</tr>
</tbody>
</table>

**Table 9: Performances**

Table 9 shows the performance for our running case study.

The execution time of searching and crawling programs take several hours: this means that explicitly programmatic control to manage failures have been designed and developed in order to manage this long-running feature and get at a result. In terms of dimension of the generated files, being it several Giga bytes it was necessary to adopt a dedicated storage platform (Apache SOLR). The usage of this platform permitted an indexed access to the generated document base.

**Open issues**

- Evaluation of accuracy of the result set for the task by considering different search engines
- Need to set up human-based controls to complete the matching task between enterprises’ names and retrieved URLs.
Pilot identification

1 NL 1 1 (first pilot implemented by Netherlands regarding use case number 1)

Reference Use case

- 1) URL Inventory of enterprises
- 2) E-commerce from enterprises’ websites
- 3) Job advertisements on enterprises’ websites
- 4) Social media presence on enterprises’ web pages

Synthesis of pilot objectives

Instead of using the ICT survey for identifying the population, in this project Statistics Netherlands used the business register (BR) for its URL inventory. The Dutch business register contains about 1.5 Million enterprises of which roughly 1/3 has a URL administered. Nothing was known about the quality of the URL field beforehand.

We took a random sample of 1000 enterprises with URL from the BR and collected information from the web using searching and scraping. 70 % of the results were used to train a model to predict the correctness of a found URL. The remaining 30 % of the results were used to validate the model. We did this in two iterations: first on a sample from the BR without any restriction on the number of employees. Second, we repeated the pilot on a sample of the BR with the restriction that the enterprise must have 10 or more employees. This approach was taken to be in line with the other countries involved in the project which all took a sample of enterprises with more than 10 employees. Within the two iterations, the search strategy, the software and the model was refined. Below, we report on the second iteration only.

Pilot details

![Diagram](image)
**General description of the flow on the logical architecture**

The URL searcher consists of using the Google search API with 5 distinct search queries. These search queries were composed of different combinations of the enterprise name, address details and the word ‘contact’. The search results, especially the “snippets” (short descriptive texts) and some additional scraping results were stored in a searchable ElasticSearch database. Feature extraction, calculating scores, tokenization, removing stop words were done with the Nodejs packages Natural and the ElasticSearch functionality. In the analysis phase a classifier was trained and validated using Scikit-learn.

**Functional description of each block**

S4SGoogleSearch: nodejs package created by Statistics Netherlands to conveniently use the Google search engine API to automatically fire search requests from a program. To use it one needs a Google API key. More information can be found on [https://github.com/SNSStatComp/S4SGoogleSearch](https://github.com/SNSStatComp/S4SGoogleSearch)

S4SRoboto: nodejs package forked from the original package “roboto” created by jculvey. The original package has a flexible architecture, native support for various backend storage, automatic link extraction, and respects the robots exclusion protocol, nofollow, noindex etc. Statistics Netherlands added some features to this package. More information can be found on [https://github.com/SNSStatComp/S4SRoboto](https://github.com/SNSStatComp/S4SRoboto)

ElasticSearch: An open source distributed, search and analytics engine. More info on [https://www.elastic.co](https://www.elastic.co)

Natural: a general natural language facility for nodejs. It supports tokenizing, stemming, classification, phonetics, tf-idf, WordNet, string similarity etc. More info on [https://github.com/NaturalNode/natural](https://github.com/NaturalNode/natural)


**Description of the technological choices**

Over the past few years Statistics Netherlands gained a lot of experience scraping the web for statistics, especially in the area of price statistics. Having used Python, R and some dedicated tools for this tasks, now the majority of scraping is performed using Nodejs (JavaScript on the server). The main reason for this was that it integrates well with the language spoken on web pages itself: JavaScript. In this ESSnet Statistics Netherlands chose to adhere to this choice, while aligning as much as possible to the methodologies used collectively by the project partners. As described above this resulted in the use of two S4S (search for Statistics) packages, which are readily available on the github of the Dutch SNS StatComp statistical computer science group (SNSStatComp).

For machine learning the situation is different. In this case it is much more important to choose a powerful machine learning library, which in our feeling can be found in the Python module scikit-learn.

The (paid) Google API was chosen because Google is the number one search engine used on the web in the Netherlands and because Statistics Netherlands used this Google API in many other projects.
Concluding remarks

Lessons learned

• Methodology: Even machine learning cannot turn garbage data into gold. It all depends on having a sound training set and this is sometimes a big problem. We found out that the tuning of the URL searcher is essential to create a valida training set for the machine learning part thereafter. Other improvements on this might be worth further exploring.

• IT: Web technologies and tools change frequently. Our experience is to take whatever is useful to do the job at hand and not to try to find the best tool for a longer period of time.

• Legal: the use of the paid Google API has no legal implications. Scraping of websites was done with the Roboto package which fully respects the robot exclusion protocol and nofollows. The data was used for this experiment only.

Open issues

In this pilot we did not apply the model to the full BR yet.
**Pilot identification**

1UK1 1 (first pilot implemented by UK regarding use case number 1)

**Reference Use case**

- [ ] 1) URL Inventory of enterprises
- [x] 2) E-commerce from enterprises’ websites
- [ ] 3) Job advertisements on enterprises’ websites
- [ ] 4) Social media presence on enterprises’ web pages

**Synthesis of pilot objectives**

At the stage of carrying out the work, the UK could not scrape data from websites without first checking the Terms and Conditions. We therefore focused on a different priority to other countries – instead of using scraped information, we investigated using *registry information* to identify business websites.

When any individual or organisation registers to hold a domain name, the name and address of that individual or organisation must be provided to a registrar. This name and address information is then usually made publically available and can be accessed via a ‘whois’ lookup. We investigated the use of this registry information in identifying business websites. We used manually-identified websites and the UK ICT survey to form training and test sets.

**Pilot details**
General description of the flow on the logical architecture

We use the enterprise names with Bing Search API to obtain sets of up to 10 suggested URLs for each business, and save this to a CSV file. We then utilize a Python script to obtain registry information for each returned URL and again save to CSV, and finally perform linking, matching and analysis.

Functional description of each block

Companies sample

We utilised a stratified sub-sample from the 2015 e-commerce survey based on the survey responses. We selected -

- 100 businesses without a website
- 100 businesses with a website who did not conduct e-commerce
- 100 businesses with a website who did conduct e-commerce

For each business, a website was manually identified.

Query Bing API

The Bing API was queried with the company name only, and the first 10 returned URLs stored as candidate domains.

Utilise a whois lookup to obtain the registrant’s address information for candidate domains

Domains can be broadly categorised into generic top-level domains – ‘gTLDs’ – such as .com, .net and .org, and country-code top-level domains – ‘ccTLDs’ – such as .co.uk.

Our primary source for registrant data for gTLDs was an ICANN-whois lookup accessed via the pywhois Python module. The ICANN-whois API was queried for all 10 URLs obtained from the bing API. The relevant registrar for the .co.uk ccTLDs is Nominet, who do not offer an API. We therefore had to manually query the nominet web application, and due to resource constraints this had to be limited to the first .co.uk URL obtained by the bing API only.

Data processing and analysis

Both the nominet whois lookup and the ICANN whois lookup return a variety of information including full postal address, name of registrant, and, in some cases, a company registration number. A whois lookup was deemed to match the business register data if the company postcode matched exactly or the company registration matched exactly.

Description of the technological choices

- We utilized the Bing API as this was easiest to set up, and offers a sufficiently high free quota (up to 1,000 queries per month for free for a limited period). The Google API appears to be ‘depreciated’.
- Most of the programming was done in Python due to convenience and in-house expertise
- We utilized py-whois API as it is the only available Python package
- The data-sets were small-scale flat files and so could be held as CSV
Concluding remarks

Lessons learned

- Methodology: Utilising registry information for the top search responses and performing a very simple exact match with information on the business register can provide a website for about 37% of businesses. This is insufficient by itself but may be useful when combined with other methods.

- It is important to consider bias in any website identification method. We found it considerably easier to find websites which conduct ecommerce (47% found) than to find websites which did not conduct e-commerce (33% found)

- IT: One challenge with this work is that we could not query APIs on the ONS network, and had to work entirely off-network with anonymised data. We are working with IT colleagues to resolve this.

Open issues

This was a very small-scale pilot, and as such the results are limited. In order to run on a larger scale we need to build a larger training set.
**Pilot identification**

1 PL 1 (first pilot implemented by Poland regarding use case number)

**Reference Use case**

1) URL Inventory of enterprises  
2) E-commerce from enterprises’ websites  
3) Job advertisements on enterprises’ websites  
4) Social media presence on enterprises’ web pages

**Synthesis of pilot objectives**

Because Polish official statistics has a business register with URL addresses, the goal of this use case was to verify the data already stored in the business register. Although not all of enterprises’ URLs are included in the business register, the decision was to validate a sample set of the URLs. The aim was to verify whether the data in the business register are the same with the results obtained by ISTAT URL Retrieval software. Therefore the decision was to analyze the output of the software – two lists were compared to see the results.

**Pilot details**
**General description of the flow on the logical architecture**

The logical architecture can be divided into three steps:

1) preparing the names for URL identification from Business Register,
2) identifying URLs with Bing API,
3) comparative analysis of URLs retrieved with Bing API with URLs stored in Business Register.

**Functional description of each block**

**URL Searcher** is a software implemented by ISTAT that use Java and Bing API to find an URL of the enterprise. The software retrieves the first ten proposed URLs and put them in a text file.

**Datasets merger** – because the output of URL Searcher and Business Register data are stored in different files, the goal was to merge them to have one dataset for comparison.

**URL Comparer** – used to compare the results of URL Searcher with data stored in Business Register.

**Description of the technological choices**

**Java** – to execute IStat URL Searcher software

**Python** – language used for Dataset Merger and URL Comparer

**Pandas** – library for Python to process and analyze the data

**Concluding remarks**

**Lessons learned**

- **Methodology**
  - In many cases URL Searcher gives references to Facebook or Twitter account of the company but they cannot be eliminated as numerous URLs in business register refer to Facebook or Twitter profile of the company. Most of the URLs that match the value stored in a business belong to large companies. Small companies are not easy to identify, especially when name is the same like “Vocational School”. Therefore we decided to conduct a test with name and city as well as name, city and street of the company taken from business register.

- **IT**
  - As the results from the URL Searcher are in semi-structured csv-like file, it is easy to process the data is to use Pandas in Python to merge and process the datasets. The scripts can be executed using pyspark.

- **Legal**
  - No legal issues as Bing public API is used.
Open issues

URL Searcher gives an opportunity to verify the current URLs in business register. However manual work is also necessary to solve problems with any duplications of the same URL identified for two different enterprises.
7.3 Use Case 2 - ECommerce

Common Approach

Four countries carried out pilots for this use case:

- Bulgaria
- Italy
- Netherlands
- UK

The approach of all countries followed the same basic outline:

1. Scrape textual content from pre-identified enterprise websites
2. Create features based on the presence or absence of words in textual content
3. Use these features and some algorithm to predict whether an enterprise is engaged in ecommerce

Details of approach and differences between countries

1. Scrape textual content from pre-identified business websites
   a. Italy scraped textual content from the entire website, while Bulgaria, Netherlands and UK scraped textual content only from the top level of the enterprise website.
   b. Sample sizes varied considerably between countries: Italy scraped 78,000 enterprise websites, Bulgaria scraped 9,909, Netherlands scraped about 1,000 while the UK scraped only 300.

2. Create features based on the presence or absence of words
   a. Different countries used the textual data to create features in different ways. Italy created a term-document matrix based on the presence of any given word in any given enterprise website. The UK used a similar approach, but limited the features to the most-common words in the corpus. Netherlands and Bulgaria instead used lists or ‘dictionaries’ of keywords, with the presence or absence of each keyword on a website constituting a keyword feature – the Netherlands manually inspected a sample of websites to identify keywords, while Bulgaria tested several sets of keywords.

3. Use these features and some algorithm to predict whether an enterprise is engaged in ecommerce
   a. Italy, Netherlands and UK all used various supervised machine learning outcomes on a randomly-selected training sample, and evaluated performance against a testing set. Italy utilised a variety of algorithms – including SVMs, Random Forests, Logistic Regression, Neural Networks, and Naive Bayes – and settled on Logistic regression and random forests, while the UK used Naive Bayes only. Bulgaria did not utilise machine-learning, and instead used a filter based on their features.
Summary of findings

- Most countries were successful in identifying at least some e-commerce websites, but all struggled to get the right balance between precision and recall. Further development of methods would be needed before arriving at robust estimates.

- A simple rules-based method for identifying features, as opposed to identifying features based on word frequency, seemed to perform reasonably well. However, all countries effectively used a ‘bag-of-words’ model – treating each word independently – and several are interested in utilising more advanced NLP-type techniques.

- Where supervised machine learning approaches were utilised, the precise technique used does not seem to make an enormous difference to the results. However, no country has investigated ‘deep learning’ type techniques, which may improve performance.
7.4 List of Pilots Use Case 2

Pilot identification

“2 BG 1” 1 (first pilot implemented by Bulgaria regarding use case number 2)

Reference Use case

- 1) URL Inventory of enterprises
- 2) E-commerce from enterprises’ websites
- 3) Job advertisements on enterprises’ websites
- 4) Social media presence on enterprises’ web pages

Synthesis of pilot objectives

The BNSI used the URLs inventory (the work done in Pilot 1 BG 1) to find out whether an enterprise is engaged in e-commerce. The main objective of this pilot was scraping the webpages of companies’ official website and then by using predefined taxonomy and key words applied to the web-site text content (text mining) it should provide an accurate prediction. In order to achieve more precise results the text mining has been done in three different ways (three versions of predictions). The predicted results then were validated manually in order to evaluate which prediction is most accurate.

Pilot details
**General description of the flow on the logical architecture**

The URL crawling-scrapping tool uses information from URLs Inventory to visit the enterprises web-pages, to predict the enterprises e-commerce URLs and to scrap a title, key words and description content of the first page of enterprise web-site or predicted first pages of the e-commerce web-sites. The information gathered by the tool was stored in the DB. The e-commerce URLs database crawling interface is used by experts to manually check and validate the correct e-commerce URL of the enterprise using the information collected by URL crawling-scrapping tool. In the analysis phase the statistical hypothesis testing was done to tests how precise and comprehensive the web-scrapping prediction algorithms were (precise means companies identified as e-traders are really e-traders; comprehensiveness is the ability of algorithm to capture all e-traders in the population of companies). In addition the final results were verified with ICT survey data. The statistical results were calculated with specific software script.

**Functional description of each block**

The URL crawling-scrapping tool takes web address from the URL Inventory (9809 URLs from Pilot 1BG1) and scrapes the enterprise Internet website first page content. Then by using PHP script with three logical algorithms (using 4 positive and 1 negative lists of key words) predicts e-commerce URL of the enterprise. The results are as follow: algorithm 1 - 1139 URLs e-commerce of enterprises; algorithm 2 – 1048 URLs e-commerce of enterprises; algorithm 3 - 662 URLs e-commerce of enterprises. The URL crawling-scrapping tool extracts the enterprise e-commerce web-address, title, key words and description of the web-site and stores the extracted information in SQL Database together with the predicted e-commerce URLs.

The e-commerce URLs database crawling interface presents results of e-commerce predictions and e-commerce’s’ first pages and enterprises first pages titles, key words and descriptions. Then the experts validated the presented results manually and real e-commerce’s URLs of enterprises were stored. In this way, we found total of 856 e-commerce web-pages.

The negative assessment is tested with statistical hypothesis (10% sample): assuming 90 % precision and 80 % comprehensiveness are OK for the algorithm. Creating a 10% sample from both populations: e-traders and remaining non e-traders and verified statistically whether precision is lower than 90 percent and comprehensiveness is lower than 80 percent. Employing normal distribution for both hypotheses we got to the conclusion that our filter is both precise and comprehensive. As a result we found that 27 e-traders were not covered from the prediction algorithms. In addition, the expected result of this pilot was to make a decision whether the produced information can be used to replace some questions contained in the questionnaire of the ICT survey.

The verification of the results is done with ICT survey data. After the benchmark analysis between ICT survey data and information obtained by the pilot we got the following results: from 26 836 enterprises (scope of the project), 4 332 enterprises were included in the ICT survey sample for 2016. We found 89 new enterprises are doing e-commerce activity, which are included in the ICT survey sample with negative answers.
Results

<table>
<thead>
<tr>
<th></th>
<th>Web scraping pilot</th>
<th>ICT sample survey</th>
<th>Success rate of WebScraping</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of enterprises</td>
<td>Number of enterprises</td>
<td>Success rate of WebScraping</td>
</tr>
<tr>
<td>e-stores – final results</td>
<td>148</td>
<td>351</td>
<td>42.17%</td>
</tr>
<tr>
<td>e-stores – first prediction algorithm</td>
<td>204</td>
<td>351</td>
<td>58.12%</td>
</tr>
</tbody>
</table>

**Figure 5: Results**

Description of the technological choices

BNSI used the same set of IT tools for this pilot as the choice in Pilot 1BG1. We used own made scripts to scrap web content and to help experts determined the correct e-commerce web sites of the enterprises.

Concluding remarks

**Lessons learned**

- Methodology: Generally, if the enterprises e-stores are there main web sites or there is a link to the e-stores from the enterprises first web pages. Sometimes, the web site creator puts link to his own e-store, so a negative list of key words with web sites creators must be used at least. Bigger list of key words not always give better results. The stricter algorithm gives more precise results, but misses more e-commerce’s URLs. The more loosed algorithm finds more web sites that are not e-stores. If we get the 856 e-commerce web pages found and get the missed 27 e-commerce from the 10% sample, then we get 1126 probable e-commerce’s URLs of enterprises (856+27*10=1126), which closes to the number of 1139 URLs e-commerce of enterprises predicted with the more loose algorithm. In total, 11.5% of enterprises (10+ employees) with websites do e-commerce and 4.2% of enterprises with 10 and more employees do e-commerce.
- IT: The used IT tools were sufficient for the Pilot.
- Legal: There were no legal issues, no law constraints on web-browsing and scrapping.

Open issues

There are no open issues in this pilot. All defined activities in the use-case were successfully executed.

Pilot identification

2 IT 1 (first pilot implemented by Italy regarding use case number 2)
Reference Use case

1) URL Inventory of enterprises
2) E-commerce from enterprises’ websites
3) Job advertisements on enterprises’ websites
4) Social media presence on enterprises’ web pages

Synthesis of pilot objectives

Given a set of enterprises’ websites, the pilot has the objective to detect whether the websites provide or not web sales facilities.

Pilot details

General description of the flow on the logical architecture

Our input is a txt file (named seed.txt) containing the list of enterprises’ websites specified as URLs. The following steps are 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 (about 78,000 out of 130,000).

**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** 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
Description of the technological choices

- We developed a set of ad-hoc Java programs, including: RootJuice and FirmDocTermMatrixGenerator.
- All of the programming was done in Java and R due to in-house expertise.
- Due to the particular domain (Big Data) we decided to use Solr that is not only a NoSQL DB but also an enterprise search platform usable for searching any type of data (in this context it was used to search web pages). In fact its major features include full-text search, hit highlighting, faceted search, dynamic clustering, database integration, rich document handling, distributed search, index replication and high scalability.
- In order to decouple the logical layers and because it is a very common and easy to manage data format, we often used csv files to store intermediate data.
- When it was possible we wrapped up already existing pieces of SW (e.g. Crawler4J)
- We used the Java library SnowballStemmer for stemming. Main reason easy multilanguage support.
- We used the library TreeTagger for lemmatization. Main reason easy multilanguage support.

Concluding remarks

Lessons learned

- Methodology: the performance of the different models used to predict values at unit level has been evaluated to be not yet satisfactory from the point of view of their capability to find true positives (though acceptable in terms of overall accuracy). For this reason, particular attention will be paid on the possibility to enrich and improve
  - the phase of web scraping (by including tags and images as inputs for next steps)
  - the phase of text processing (by using Natural Language Processing techniques to consider not only single terms but n-grams of terms)
  - the phase of machine learning (by considering new learners derived from Deep Learning).
- IT: we decided to decouple the scraper and the storage platform for both performance and sustainability reasons. Indeed, in terms of performance we experimented technical problems in dealing with Solr Connection pool in the loading phase right after the scraping one. In terms of sustainability, given that we don’t have yet an enterprise level platform for document databases in Istat, we decoupled from Solr, leaving open the possibility of using another similar solution (e.g. elastic Search).
- Legal. We are currently working on the final version of the agreement with our National Authority for Privacy especially to point out the measures for protection of personal data possibly involved in the scraping task.

Open issues

- Evaluation of the scraped result “stability”: it is relevant to point out that different runs of the scraping system may produce different result set; it is relevant to assess the impact of these differences on the subsequent analysis task.
- Degree of independence of the access and data preparation layers from analysis approaches. Though the two layers have been designed and developed with very general requirements in mind, it might be the case that not a full independence of the analysis layer from them has been achieved. This could result in minor changes to be performed on the scraping and data preparation software applications.
Pilot identification

2 NL 1 (first pilot implemented by Netherlands regarding use case number 2)

Reference Use case

1) URL Inventory of enterprises
2) E-commerce from enterprises’ websites
3) Job advertisements on enterprises’ websites
4) Social media presence on enterprises’ web pages

Synthesis of pilot objectives

Instead of using the population of the ICT survey, in this project Statistics Netherlands used a list of about 1200 foreign companies paying Dutch VAT to the Dutch tax authorities. This set was chosen because there was a clear demand from the statistical division to automatically determine web sales activities in the Netherlands for these enterprises.

In this pilot we re-used as much as possible the work done in the URL inventory pilot (Pilot 1 NL 1). The set of foreign enterprises was fed into the URL finding software which resulted in a set of URLs with an indication of its correctness (good, fair, mediocre, poor). Based on this a set of about 1000 sites were crawled using dedicated crawling software. Automatic text analysis resulted in an indication of web shop activities with an accuracy of about 85 %.

Pilot details
General description of the flow on the logical architecture

For a more detailed description of the URL finding step we refer to the description of pilot 1 NL 1. After finishing this operation on about 1000 enterprises a number of websites resulting from the search step were inspected. It turned out that all of them had some kind of indication of a shopping facility on their home page. For each enterprise the first page was scraped and analysed. A simple deterministic approach was used to extract features: by analysing several sites (including their technical design) a set of keywords was designed that indicate whether the site executes webshop activities in the Netherlands or not. Note that these sites were owned by non-Dutch companies, so the set of keywords had to be designed including terms from other non-Dutch languages such as English, German, French and Turkish. After validating the set of keywords on a subset of sites, the information extraction step was executed on the whole set.

Functional description of each block

Scrapy: is a well-known general purpose scraping library in Python. See https://scrapy.org

Custom Python: for feature extraction based on a set (vocabulary) of terms that were designed after the analysis of a subset of sites in different languages.

Custom R: The execution of all steps on the complete set of 1200 enterprises was carried out in R.

Description of the technological choices

See pilot 1 NL 1. In addition to the architecture and technologies used in the URL finding step, in the consecutive steps Python and R were used for the feature and information extraction.

Concluding remarks

Lessons learned

- Methodology: A fairly simple deterministic approach on detecting E-commerce activities on a set of sites of enterprises proofed to be successful. Improvements could be to have language-specific vocabularies.
- IT: Web technologies and tools change frequently. Our experience is to take whatever is useful to do the job at hand and in this pilot this led to the use of Python and R.
- Legal: For URL finding see pilot 1 NL 1. With respect to the additional scraping for E-commerce using Scrapy we note that this package fully respects the robot exclusion protocol etc. With respect to this specific experiment one could ask is it makes a difference to scrape sites from one country owned by enterprises located in another country. For this experiment we did not dive into that any further.

Open issues

None
**Pilot identification**

2UK1 (first pilot implemented by UK regarding use case number 1)

**Reference Use case**

- 1) URL Inventory of enterprises  
- X 2) E-commerce from enterprises’ websites  
- 3) Job advertisements on enterprises’ websites  
- 4) Social media presence on enterprises’ web pages

**Synthesis of pilot objectives**

Our aim was to scrape data from a sample of businesses which appear in the UK ICT survey and train a supervised machine learning model, using survey responses as training data, to predict whether a business is engaged in ecommerce. We scraped only the top level of a website and used a simple bag-of-words Naive Bayes classifier.

The main difference from other pilots in the use case is that, for business websites, we manually identified websites rather than using the output of use-case 1 or an administrative source.

**Pilot details**
**General description of the flow on the logical architecture**

A sample from e-commerce survey was taken and websites manually identified for these businesses. For each business, the top level of the website was scraped using scrapy, with results stored in a text file. A custom python script is then used to create features and train a Naive Bayes classifier to predict whether a business is engaged in e-commerce.

**Functional description of each block**

**Companies Sample**

From respondents to the UK ICT survey who reported having a website, 100 businesses who report conducting e-commerce on their website and 100 who do not were sampled. For each of these businesses, a website has been manually identified.

**Scrape using Scrapy**

We attempted to scrape every website using Scrapy. Where scraping is blocked through the robots.txt exclusion protocol or otherwise, no scraping took place. Due to this restriction, and the fact that we could not find a website for some businesses, we successfully scraped data for only 140 enterprises.

**Feature Extraction**

The features for our Naive Bayes classifier were simply the presence or non-presence of words on a website, with each word treated independently (‘bag-of-words’ model). Features were created for the 5000 most common words across all websites in the study.

**Machine Learning**

We used a simple Naive Bayes classifier, with an 80%-20% training/testing split.

**Evaluation**

We computed standard accuracy metrics on our test data.

**Description of the technological choices**

Scraping using scrapy – scrapy is a powerful, scalable open-source web-scraping tool available in Python. We chose scrappy because we have used it in other projects, and it seemed to be a good fit here.

Feature extraction and machine learning using the NLTK library in python. This is a powerful natural-language toolkit with native functionality for machine learning which would easily allow more advanced natural language techniques.

Other technological choices - see Use Case 1.
Concluding remarks

Lessons learned

- Methodology – a simple naive bayes classifier using a bag of words model can have reasonable accuracy (71.4%). However, we believe significant improvements could be made from using more complex natural language processing.
- Methodology – a much larger training set is required.
- IT – scrapy is a useful, scalable tool for web-scraping. The NLTK library in python will be useful for investigating more advanced natural language processing.
- Legal – as a part of this pilot we needed to address legal issues within the ONS. We now have provisional guidance which allows us to web-scrape, respecting the robots.txt protocol, without checking the terms and conditions of websites.

Open issues

None
7.5 Use Case 3 – Job Advertisements

Approach

Three countries carried out pilots for this use case:

- Italy
- Sweden
- UK

The basic outline:

1. Scrape the web pages by a list of enterprise URLs.
2. Identify job advertisements.
3. Evaluate the result.

Details of approach and differences between countries

1. Scraping web pages
   a. Italy scraped the web content with the URLs provided.
   b. Sweden scraped the first web page content with 100 URLs of the enterprises in the public sector; 282 pages were scraped.
   c. UK scraped 400 websites, of which 50% contained job advertisements and 50% contained no job advertisements; 309 pages were scraped in the end.

2. Identifying job advertisements
   a. Machine learning were applied in all countries. Sweden and UK used the most common words on the web sites as features. Italy used correspondence analysis and importance in generating random forests for features selection.

3. Evaluating result
   a. Italy used accuracy and F1 measure for evaluation. Sweden conducted manual evaluation. Relative many false positive were identified and can cause overestimation of the number of vacant jobs. UK computed the standard accuracy metrics.

Summary of findings

- Python libraries scrapy and nltk are powerful for web scraping and natural language analysis.
- We need to scrape deeper in the web structure in order to identify job advertisements and even scrape images.
- One common problem is the false positives, we should try n-grams and better feature engineering methods to distinguish text about the work place from the job advertisements.
- We need to systematically build up good training sets at large scale.
- Scraping at large scale is regulated differently in different countries at the moment, the legal situations need to be clarified.
7.6 List of Pilots Use Case 3

Pilot identification

3 IT 1 (first pilot implemented by Italy regarding use case number 3)

Reference Use case

- [ ] 1) URL Inventory of enterprises
- [x] 2) E-commerce from enterprises’ websites
- [x] 3) Job advertisements on enterprises’ websites
- [ ] 4) Social media presence on enterprises’ web pages

Synthesis of pilot objectives

Given a set of enterprises’ websites, the pilot has the objective to detect whether the websites provide or not job advertisements.

Pilot details

General description of the flow on the logical architecture

Our input is a txt file (named seed.txt) containing the list of enterprises’ websites specified as URLs. The following steps are 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"). In this case the dependent variable is “Job advertisement (yes/no)” and the predictors have to selected in scraped text. This activities have been carried out:
  o 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)
  o Partitioning of data in a train and test sets equally balanced
  o 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.

**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** 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**

• 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

**Description of the technological choices**

- We developed a set of ad-hoc Java programs, including: RootJuice and FirmDocTermMatrixGenerator.
- All of the programming was done in Java and R due to in-house expertise.
- Due to the particular domain (Big Data) we decided to use Solr that is not only a NoSQL DB but also an enterprise search platform usable for searching any type of data (in this context it was used to search web pages). In fact its major features include full-text search, hit highlighting, faceted search, dynamic clustering, database integration, rich document handling, distributed search, index replication and high scalability.
- In order to decouple the logical layers and because it is a very common and easy to manage data format, we often used csv files to store intermediate data.
- When it was possible we wrapped up already existing pieces of SW (e.g. Crawler4J)
- We used the Java library SnowballStemmer for stemming. Main reason easy multilanguage support.
- We used the library TreeTagger for lemmatization. Main reason easy multilanguage support.

**Concluding remarks**

**Lessons learned**

- Methodology: the performance of the different models used to predict values at unit level has been evaluated to be not yet satisfactory from the point of view of their capability to find true positives (though acceptable in terms of overall accuracy). For this reason, particular attention will be paid on the possibility to enrich and improve
  - the phase of web scraping (by including tags and images as inputs for next steps)
  - the phase of text processing (by using Natural Language Processing techniques to consider not only single terms but n-grams of terms)
  - the phase of machine learning (by considering new learners derived from Deep Learning).
- IT: we decided to decouple the scraper and the storage platform for both performance and sustainability reasons. Indeed, in terms of performance we experimented technical problems in dealing with Solr Connection pool in the loading phase right after the scraping one. In terms of sustainability, given that we don’t have yet an enterprise level platform for document databases in Istat, we decoupled from Solr, leaving open the possibility of using another similar solution (e.g. elastic Search).
- Legal. We are currently working on the final version of the agreement with our National Authority for Privacy especially to point out the measures for protection of personal data possibly involved in the scraping task.

**Open issues**

- Evaluation of the scraped result “stability”: it is relevant to point out that different runs of the scraping system may produce different result set; it is relevant to assess the impact of these differences on the subsequent analysis task.
• Degree of independence of the access and data preparation layers from analysis approaches. Though the two layers have been designed and developed with very general requirements in mind, it might be the case that not a full independence of the analysis layer from them has been achieved. This could result in minor changes to be performed on the scraping and data preparation software applications.
**Pilot identification**

3 SE 1 (first pilot implemented by Sweden regarding use case number 3)

**Reference Use case**

- 1) URL Inventory of enterprises
- 2) E-commerce from enterprises’ websites
- 3) Job advertisements on enterprises’ websites
- 4) Social media presence on enterprises’ web pages

**Synthesis of pilot objectives**

In order to identify job advertisements on enterprises’ websites, we explored a small sample of urls. However with the restrictions of the legal situation at Stat Sweden, we could only test on a small sample of urls taken from the Swedish Employment Agency. The url sample covers only the public sector. The main objective is to gain experience before experimenting on a large scale.

**Pilot details**
**General description of the flow on the logical architecture**

Three modules are used in the pilot test, SCBScraper, SoupCleaner and Classifier. SCBScraper is used to retrieve the web content with a given url following the netiquette. The SoupCleaner cleans the html text retrieved; it then extracts and prepares the information needed for the Classifier and the SCBScraper. The Determinator(Classifier) was developed in a previous project\(^9\). It was trained with job advertisements text and non-ads webpages to determine if a webpage describes a job advertisement or not. The classifiers were fitted by Gaussian Naïve Bayes, Decision tree and Multilayer perceptron.

The pilot test was done on 100 urls of enterprises in the public sector. 21 urls contain errors and therefore were not accessible.

Of those urls that are accessible, the first pages of the urls are extracted. The links on the first webpages are extracted and analyzed. Links that contain either of the two words “jobb” or “arbeta” are considered potential links to the job advertisement. The scraper then retrieves the content from these potential links and the content is cleaned and prepared for the classifier. We prepared two corpuses from the websites scraped, one with all the text and one without the link text. The corpuses are sent to classifiers for classification. The result of the classification is saved for manually evaluation.

**Functional description of each block**

SCBScraper: this module retrieves html text by a given url. It makes sure the retrieving follows the netiquette. The netiquette includes checking if ‘robot.txt’ exists and follows the rules stated there; telling who is scraping the website; idling between the scraping. It fixes some errors of urls. It checks as well if a link follows a certain pattern and retrieves the link website if it does.

SoupCleaner: this module works on the web content retrieved. It extracts links from a website. It retrieves values of attributes in html text e.g., the ‘labels’ and the link-address (‘href’). It can save the html text without links. It calls the function “bag-of-words” and transforms the html text into a bag of words for text classification.

Determinator (Classifier): this module loads the training data and trains the classifiers with Gaussian Naïve Bayes, Decision tree and Multilayer perceptron. It saves the classifiers that can be called for classifying on new data.

**Description of the technological choices**

The pilot is carried out at SCB inlab, which is an open environment separated from the ordinary SCB environment. The modules are developed in Python. Packages used are urlibs3, urlib, BeautifulSoup, sklearn, tensorflow, pandas and other standard packages.

---

\(^9\) [http://www1.unece.org/stat/platform/display/bigdata/Report%3A+Enterprise+Web+sites](http://www1.unece.org/stat/platform/display/bigdata/Report%3A+Enterprise+Web+sites)
Concluding remarks

The pattern analysis on the links from the first url page returned the job relevant links. Since it is designed as a pattern search, it is easy to adapt to new patterns.

The classifier although returned correct answers, the result cannot be used for evaluation the method. Since we scraped only two-levels along the links from the first urls given, the pages are mostly non-job advertisements. However experiences are gained and lessons learned are valuable. It is highly possible to identify job advertisements from the websites.

Lessons learned

• Method:
  i. The classifiers return many false positive results on several kinds of pages, e.g. pages that describe the enterprise as employer; pages describing the work conditions and about projects. Even internal job lists are classified as job advertisements. These pages are related closely to the advertisements, which might need to be categorized separately;
  ii. The test was done on two corpus of text, one with the links’ text and one without the link’s text. It is important to remove the irrelevant text such as the menu and framework of the website and extract only the relevant text for text classification. In our test the few job advertisements are identified correctly by the text without links’ text and failed on the text with the links’ text;
  iii. We should go deeper along the potential job links and retrieve more pages for analysis. The assumption is that if enterprises publish job ads on their websites, they can be retrieved if we go to the end of the links;
  iv. De-duplicating the pages is needed since the same pages are retrieved from different links;
  v. We do not see advantages and disadvantages of one classifier to another; we can test several more, e.g. support vector space, and combining the results to see if it improves the result.

• Legal: it is a good start with the analysis on the sample of websites from the public sector and the web scraping netiquette. If we are to continue with a larger sample, the legal situation needs first to be clarified. An alternative approach (tested in work package 1) is to seek cooperation with main job portals, as we then are able to obtain data without web scraping.
Pilot identification

3UK1 (first pilot implemented by UK regarding use case number 3)

Reference Use case

☑ 1) URL Inventory of enterprises ☐ 2) E-commerce from enterprises’ websites
☑ 3) Job advertisements on enterprises’ websites ☐ 4) Social media presence on enterprises’ web pages

Synthesis of pilot objectives

Our aim was to scrape data from a sample of businesses websites – 50% of which contain job adverts – and use the scraped data to train a supervised machine learning model predicting the presence of job adverts. This was a small-scale pilot to help us explore technological, methodological and ethical issues with carrying out this kind of web-scraping in a UK context.

Pilot details
**General description of the flow on the logical architecture**

We attempted to scrape data from 400 business websites, 200 of which contained job vacancies. The scraping was carried out using Scrapy with results stored in a text file for each website. We then used a custom python script to create features based on the presence of words on a website and trained a Naive Bayes classifier to predict whether a business website contains job vacancies.

**Functional description of each block**

**Companies Sample**

Unfortunately, the UK ICT survey does not capture information on whether a business advertises job vacancies online. We obtained ‘positive’ (websites with job vacancies) and ‘negative’ (websites without job vacancies) samples separately:

- **Sample of websites not containing job vacancies:** we sampled businesses from the UK ICT survey which report having a website, manually identify a website for each of these businesses, and verified that this website does not contain job vacancies.

- **Sample of websites containing job vacancies:** these websites were scraped from an online listings page containing business websites which offer jobs vacancies. This listings page does not exclude scraping in the terms and conditions.

Our final sample contained 400 business websites – 200 containing job vacancies and 200 which don’t.

**Scrape using Scrapy**

We attempted to scrape every website using Scrapy. Where scraping is blocked through the robots.txt exclusion protocol or otherwise, no scraping took place. Due to these restrictions we successfully scraped data for only 309 enterprises.

**Feature Extraction**

The features for our Naive Bayes classifier were simply the presence or non-presence of words on the website, with each word treated independently (‘bag-of-words’ model). Features were created for the 50,000 most common words across all websites in the study.

**Machine Learning**

We used a simple Naive Bayes classifier, with an 80%-20% training/testing split.

**Evaluation**

We computed standard accuracy metrics on our test data.
Description of the technological choices

Scraping using scrapy – scrapy is a powerful, scalable open-source web-scraping tool available in Python. We chose scrapy because we have used it in other projects, and it seemed to be a good fit here.

Feature extraction and machine learning using the NLTK library in python - This is a powerful natural-language toolkit with native functionality for machine learning which would easily allow more advanced natural language techniques.

Other technological choices - see Use Case 1.

Concluding remarks

Lessons learned

- Methodology – for the main pilot we only scraped the top level of the website. Our other investigations suggest that this is insufficient – to capture jobs listing data, a fairly deep crawl is required.

- Methodology - we had a particular problem with False Positives – websites being identified as containing websites when they actually do not. Some of the most informative features were words like ‘opportunities’, which may appear on most business websites. We used a simple ‘bag of words’ naive bayes classifier, and it may be that using more complex natural language processing methods would address this False Positive issue.

- Methodology – a much larger training set is required. We are working to build up our URL inventory to address this issue.

- IT – scrapy is a useful, scalable tool for web-scraping. The NLTK library in python will be useful for investigating more advanced natural language processing.

- Legal – as a part of this pilot we needed to address legal issues within the ONS. We now have provisional guidance which allows us to web-scrape, respecting the robots.txt protocol, without checking the terms and conditions of websites.

Open issues

We need a much larger training set, and will re-do this research using more data when our expanded URL inventory is available. At this time we will crawl more pages from each website.
7.7 Use Case 4 – Social media presence on enterprises’ web pages

Approach

Three countries carried out pilots for this use case:

- Bulgaria
- Italy
- Poland

The basic outline:

1. Scraping the web pages with a list of enterprises’ URLs.
2. Identifying accounts of the social media (mainly Facebook and Twitter).
3. Evaluating the result.
4. Scrape the social media data for usage analysis

Details of approach and differences between countries

1. Scraping web pages
   a. Bulgaria scraped the first web page content with the urls from the URL inventory; Poland went deeper on websites if anchors were not found on the first page until all the internal links were checked; Italy scraped all the pages according to the url list.

2. Identifying the social media accounts
   a. Bulgaria checked links on the pages scraped and filtered with the social media names; Italy applied machine learning and used a subset of 2016 ICT survey answers; Poland identified links to social media.

3. Evaluating result
   a. Bulgaria and Italy evaluated the result with ICT survey data.

4. Social media usage analysis
   a. Poland conducted this step on Twitter data and applied sklearn machine learning studying the purpose of the social media presence

Summary of findings

- Most countries were able to identify social media accounts. In Bulgaria, the accounts found by web scraping are less than the ICT survey answer.
- One common problem is that social media accounts identified are not necessarily corresponding to the urls or companies searched for. It is not sure either if the accounts are updated or not.
- The analysis of the social media data with Twitter API is efficient but difficult with the current Facebook API.
- Scraping at large scale is regulated differently in different countries at the moment.
7.8 List of Pilots Use Case 4

Pilot identification

“4 BG 1” (first pilot implemented by Bulgaria regarding use case number 4)

Reference Use case

- 1) URL Inventory of enterprises
- 2) E-commerce from enterprises’ websites
- 3) Job advertisements on enterprises’ websites
- 4) Social media presence on enterprises’ web pages

Synthesis of pilot objectives

The BNSI used the URLs inventory (the work done in Pilot 1BG1) to check whether an enterprise is presented or not in the social media (facebook, twitter, linkedin, google, youtube, pinterest, Instagram). The main objective of this pilot is to provide information on the activity of Bulgarian companies in the social media. It means all social media accounts will be taken into account. The general concept of this use case is to scrap the webpage and search for any links to social media accounts mentioned above. In the future more attributes can be added e.g., whether the account is up to date and how often the content is changing. The expected result of this pilot was to make a decision whether the data on social media activity can be used to keep business registers up to date and whether Bulgarian company presence in social media. In addition, the expected result could be used to decide on replacing some questions contained in the ICT survey.

Pilot details

![Diagram of internet access, storage, data preparation, and analysis for the pilot.](image-url)
**General description of the flow on the logical architecture**

The URL crawling-scraping tool uses information from URLs Inventory to visit the enterprises web-pages, to predict the enterprises presence on social media. The information gathered by the tool was stored in the DB. In addition the final results were verified with ICT survey data. The statistical results were calculated with specific software script.

**Functional description of each block**

The URL crawling-scraping tool (the same tool used under the Pilot 2BG1) takes web address from the URL Inventory (9809 URLs from Pilot 1BG1) and scrapes the enterprise Internet website first page content. Then by using the social media name the PHP script is looking for a web-link to the social media profile and stores the founded information in SQL Database. The obtained results are as follows: facebook - 2356 profiles, twitter – 922 profiles, linkedin - 560 profiles, google - 871 profiles, youtube - 527 profiles, pinterest – 139 profiles, Instagram – 127 profiles. 24.9% of enterprises (10+ employees) with websites have at least one social media profile and 9.1% of enterprises with 10 and more employees use at least one of the covered social media.

We tested results of web scraping for precision and comprehensiveness by means of statistical test of hypothesis. We made a 10% sample and statistically tested whether precision is 90 % and comprehensiveness is 80 %. More concretely we tested whether precision is lower than 90 % and comprehensiveness is lower than 80 %. Employing normal distribution for the zero hypothesis we got to the conclusion that our filter is both precise and comprehensive. As a result we found that 4 enterprises were not covered. In addition, the expected result of this pilot was to make a decision whether the produced information can be used to replace some questions contained in the questionnaire of the ICT survey.

The verification of the results is done with ICT survey data. After the benchmark analysis between ICT survey data and information obtained by the pilot we got the following results: from 26 836 enterprises (scope of the project), 4 332 enterprises were included in the ICT survey sample for 2016. We found 382 new enterprises are presented on social media which are included in the ICT survey sample with negative answers.

<table>
<thead>
<tr>
<th></th>
<th>Web scraping pilot</th>
<th>ICT sample survey</th>
<th>Success rate of WebScraping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Social media</td>
<td>1177</td>
<td>3144</td>
<td>37.44%</td>
</tr>
</tbody>
</table>

**Figure 6: Results**

**Description of the technological choices**

BNSI used the same set of IT tools for this pilot as the choice in Pilot 1BG1. We used own made scripts to scrap web content and to help experts determined the correct e-commerce web sites of the enterprises.
Concluding remarks

Lessons learned

• Methodology: The risk is that some of the Facebook or Twitter links presented on webpage may be linking to other enterprises. This is the reason that evaluation of Facebook and Twitter profiles is necessary to provide reliable information. In some cases, enterprises can have several Facebook profiles. Therefore, it is necessary to link the main profile to the company. Generally, the social media links to the enterprises profiles are situated on the first pages of the enterprises web sites.
• IT: The used IT tools were sufficient for the Pilot.
• Legal: There were no legal issues, no law constraints on web-browsing and scrapping.

Open issues

There are no open issues in this pilot. All defined activities in the use-case were successfully executed.
Pilot identification

4 IT 1 (first pilot implemented by Italy regarding use case number 4)

Reference Use case

- 1) URL Inventory of enterprises
- 2) E-commerce from enterprises’ websites
- 3) Job advertisements on enterprises’ websites
- 4) Social media presence on enterprises’ web pages

Synthesis of pilot objectives

Given a set of enterprises’ websites, the pilot has the objective to detect for each website the presence of the enterprise on the social media.

Pilot details

General description of the flow on the logical architecture

Our input is a txt file (named seed.txt) containing the list of enterprises’ websites specified as URLs. The following steps are 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"). In this case the dependent variable is “Presence in social media (yes/no)” and the predictors have to selected in scraped text. This activities have been carried out:
  o 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)
  o Partitioning of data in a train and test sets equally balanced
  o 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.

**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** 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

**Description of the technological choices**

• We developed a set of ad-hoc Java programs, including: RootJuice and FirmDocTermMatrixGenerator.
• All of the programming was done in Java and R due to in-house expertise.
• Due to the particular domain (Big Data) we decided to use Solr that is not only a NoSQL DB but also an enterprise search platform usable for searching any type of data (in this context it was used to search web pages). In fact its major features include full-text search, hit highlighting, faceted search, dynamic clustering, database integration, rich document handling, distributed search, index replication and high scalability.
• In order to decouple the logical layers and because it is a very common and easy to manage data format, we often used csv files to store intermediate data.
• When it was possible we wrapped up already existing pieces of SW (e.g. Crawler4J)
• We used the Java library SnowballStemmer for stemming. Main reason easy multilanguage support.
• We used the library TreeTagger for lemmatization. Main reason easy multilanguage support.

**Concluding remarks**

**Lessons learned**

• Methodology: the performance of the different models used to predict values at unit level has been evaluated to be not yet satisfactory from the point of view of their capability to find true positives (though acceptable in terms of overall accuracy). For this reason, particular attention will be paid on the possibility to enrich and improve
  o the phase of web scraping (by including tags and images as inputs for next steps)
  o the phase of text processing (by using Natural Language Processing techniques to consider not only single terms but n-grams of terms)
  o the phase of machine learning (by considering new learners derived from Deep Learning).
• IT: we decided to decouple the scraper and the storage platform for both performance and sustainability reasons. Indeed, in terms of performance we experimented technical problems in dealing with Solr Connection pool in the loading phase right after the scraping one. In terms of sustainability, given that we don’t have yet an enterprise level platform for document databases in Istat, we decoupled from Solr, leaving open the possibility of using another similar solution (e.g. elastic Search).
• Legal. We are currently working on the final version of the agreement with our National Authority for Privacy especially to point out the measures for protection of personal data possibly involved in the scraping task.
Open issues

- Evaluation of the scraped result “stability”: it is relevant to point out that different runs of the scraping system may produce different result set; it is relevant to assess the impact of these differences on the subsequent analysis task.
- Degree of independence of the access and data preparation layers from analysis approaches. Though the two layers have been designed and developed with very general requirements in mind, it might be the case that not a full independence of the analysis layer from them has been achieved. This could result in minor changes to be performed on the scraping and data preparation software applications.
Pilot identification

4 PL 1 (first pilot implemented by Poland regarding use case number 4)

Reference Use case

☐ 1) URL Inventory of enterprises
☐ 2) E-commerce from enterprises’ websites
☐ 3) Job advertisements on enterprises’ websites
☒ 4) Social media presence on enterprises’ web pages

Synthesis of pilot objectives

The use case was divided into two parts:

1) Webscraping of enterprise websites and searching for any anchors to social media that appeared on the website. If it is not on the main page, the software allows to go deeper into website and identify links. The result of this part is a CSV file that contains links to all social media portals found on the website.

2) Webscraping of Twitter data – if Twitter accounts were identified a Machine learning algorithm is trying to identify what is the purpose of social media presence.

Pilot details

General description of the flow on the logical architecture

The URL list is taken from the Business Register called Statistical Units Database. Then the scraper is going through the website to find a social media link. If the link cannot be found, the scraper may go
deeper to visit all internal links. In the second stage a training dataset is used to identify the type of social media activity on Twitter.

**Functional description of each block**

Each URL taken from the business register database is stored in CSV file. Then the URL is scraped to find the possible social media presence. The website is not stored – it is processed in the memory. The HTML Parser is extracting all links and possible hyper links. The links are divided into internal (for deeper analysis if no social media link is on the main web page) and external (to find social media portals). All the links found on the main page (or sub pages if there is a need for deep analysis) are stored in CSV files as a collection divided into different social media channel. In the second step Twitter links are taken from CSV file and checked with sklearn machine learning library to find the purpose of the social media presence of the enterprise. The training dataset has been prepared based on real Twitter accounts of enterprises.

**Description of the technological choices**

Language:

- Python 3 (there are difference between Python 2 and 3 – both of them are still developed – current versions 2.7 and 3.6)

Libraries:

- Tweepy (Twitter API for Python)
- BeautifulSoup4 (HTML Parser for Python)
- Sklearn

Platform:

- Apache Spark/Hadoop (to execute Python scripts)

**Concluding remarks**

**Lessons learned**

- Methodology
  - Sometimes there are more than one account on one social media portal (e.g., two accounts on Twitter). It means that enterprise is using accounts for different purposes, e.g., promotions of specific product.
  - The best way is to find the anchor to the specific social media channel (e.g., on Facebook, Twitter or Youtube as it is mentioned in the survey ICT in enterprises). However in some cases we need to search through the websites to find hidden links.
  - When the link cannot be found on the main page, there is a need to go deeper into the website. In this specific use case we can set the level to which we want to go deeper. The level must be set to modest number because going deeper may slow the analysis and also the robot can be blocked.
  - Some of the social media addresses may refer to different companies – if the partner company has only Facebook fan page as their website.
- Machine learning SVM algorithm has better accuracy in identifying the type of the comment than NaiveBayes. However there is a need to unify methods of text mining and machine learning for the whole process.
- We need a big training sets to have a good accuracy.

**IT**
- We don’t need efficient tools as web scraping and finding a specific link does not consume too much time.
- If we want to look into Facebook accounts of enterprises – it will be quite difficult with the current Facebook API.
- Twitter API is very efficient but in free version there is a limit of the number of queries in a specific period of time.

**Legal**
- Before web scraping it is recommended to see in robots.txt file if the owner of the website agreed to do such tasks.
- Still massive web scraping is not regulated.

**Open issues**

- There is a need to the discussion if we can supplement ICT in Enterprises with additional more detailed information on social media activities.