Sustainability Reports

Alessandra Sozzi

Submitted for the Degree of Master of Science in

Data Science and Analytics (Year in Industry)

Department of Computer Science
Royal Holloway University of London
Egham, Surrey TW20 0EX, UK

June 16, 2014

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count: 14,409

Student Name: Alessandra Sozzi

Date of Submission: September 14, 2016

Signature: Alessandra Sozzi

Abstract

Nowadays the Web represents a medium through which corporations can effectively disseminate and demonstrate their efforts to incorporate sustainability practices into their business processes.

This led to the idea of using the Web as a source of data to measure how UK companies are progressing towards meeting the new sustainability requirements recently stipulated by the United Nations.

The main challenges associated with this project come from not knowing a priori the structure of companies' websites and where to find the relevant information. As with any data acquired from the Web, the format was messy, cluttered with additional features of no interest and sometimes was not what we were looking for at all.

This report describes the steps taken to the development of a web scraping program able to collect sustainability information from websites of a sample of 100 companies and the use of natural language processing techniques to process and validate the data collected.

The results show that it is possible to discern the number of companies publishing sustainability information via scraping of their websites.

However, the mere action of searching for keywords might not be sufficient in the context of Official Statistics, and here it is why this project goes a step further, applying text analysis to the content of the page to extract additional insights.

Contents

1 Introduction

In Sept 2015 the United Nations stipulated its requirements for Sustainable Development Goals (SDG's)[1].

The SDGs outline a blueprint for development priorities to be achieved by 2030.

They include 17 goals with 169 corresponding targets covering a broad range of sustainable development issues, which span from ending poverty and hunger to improving health and education, reducing inequality, and combating climate change.

The Goals are being followed-up and reviewed using a set of global indicators, which result from the aggregation of the national levels indicators produced by each member state.

This is a high profile work which received strong backing from the UK Prime Minister[2].

The Office for National Statistics (ONS) has the responsibility for reporting the UK's progress towards the SDGs.

To accomplish this, an SDG team has been formed within ONS and tasked with coordinating the production of estimates to answer these indicators for the UK. The SDG team have pointed out that some of the indicators can be met by using existing ONS outputs or data. However, there are several indicators which currently ONS cannot provide estimates for or cannot disaggregate to the required level. To this end, the SDG team have asked the ONS Big Data team to provide some support around the potential use of alternative data sources for producing those missing.

The aim of this report is to outline an initial proof-of-concept research for the indicator "Number of companies publishing sustainability reports, by turnover band, geography, national or global company, sector and number of employees" [1] using an alternative Web-based data source.

This indicator relates to SDG Target 12.6 which is to encourage companies, especially large and transnational companies, to adopt sustainable practices and to integrate sustainability information into their reporting cycle [1].

The idea proposed here is to use companies' websites as a source, letting a web scraping program collect any sustainability information published on these websites.

Along the development of the project, several issues have been encountered, as expected when dealing with real world data. Challenges and opportunities that go with this choice are outlined in this report.

At first, the rationale and research objectives are stated.

---

[1] At the United Nations Sustainable Development Summit on 25 September 2015, world leaders adopted the 2030 Agenda for Sustainable Development, a plan of action for people, planet and prosperity. For more information see `https://sustainabledevelopment.un.org`

[2] PM's speech to the UN Sustainable Development Goals Summit 2015. See `https://www.gov.uk/government/speeches/pms-speech-to-the-un-sustainable-development-goals-summit-2015`

Given that the refinement of the SDG indicators is a work in progress, the Definitions Section helps to outline the assumptions made at the start of the project.

Section 2 is devoted to the web scraping application, and can be decomposed into three main parts:

1. The acquisition of the list of companies to be scraped
2. A description of the framework used to develop the application and how the several pieces fit together in the web scraping process
3. A focus on the extraction of the text from web pages and how the content extracted is then saved to a database.

Follow some statistics on the content scraped, which will give some initial insights. Section 4 illustrates the preprocessing of the text, a fundamental task to be performed before moving to the application of the Latent Dirichlet Allocation (LDA), a topic modelling technique explained in Section 5. The analysis of the results and an assessment of the overall project are ultimately exposed. The report concludes with reflections on the ethical meaning of web scraping and a section listing recommendations for future steps.

1.1 Project Rationale and Objectives

The objectives defined at the beginning of this project were to:

1. Develop a web scraping program able to collect sustainability information from companies' websites
2. Use text analysis to extract insights from the data collected
3. Make recommendations about the suitability of the methods and resulting data for developing SDG indicators

A secondary aim was to show how the Internet can be utilised as a data source and be harnessed in substitution of data gathered using traditional instruments of statistical surveys, a mission the ONS Big Data team is committed to. This brings to the two research questions originally identified for this project:

- Is the web scraping approach usable?
- To what extent the data gathered fulfils the indicator?

Future work will require liaising with the SDG team to formally evaluate the approach taken, refine some of the requirements and eventually extend the development of the application to run in a production environment to provide a wide-coverage estimate for the indicator.

1.2 Definitions

As announced earlier in the introduction, this project was developed to measure companies' response to the SDG's call for action. In particular, the request from the SDG team was made with regards to:

> **Target 12.6** To encourage companies, especially large and transnational companies, to adopt sustainable practices and to integrate sustainability information into their reporting cycle.
> Target 12.6 falls under the general Goal 12:

Ensure sustainable consumption and production patterns.

According to the UN definition "Sustainable consumption and production is about promoting resource and energy efficiency, sustainable infrastructure, and providing access to basic services, green and decent jobs and a better quality of life for all" [2].

Long-term interests of an organisation are best served by improving its economic, sociocultural, and environmental/energy practices, also known as the three pillars of sustainability defined in the 2005 World Summit on Social Development [3].

Corporate social responsibility (CSR), or often Sustainability, are the words often used to indicate the voluntary commitment of a business to includes the three pillars in its corporate processes. The first assumption made is that these two terms are interchangeable, and can both be used to indicate a company's dedication to any of the sustainability aspects just mentioned: anything that goes from resource efficiency, environment protection, to assisting the local community, to improving worker life, etc.

For the purpose of this research, the second assumption made is that companies exploit the Web and will seek to do it more and more in the foreseeable future to make sustainability information widely visible and help reinforce the relationship between their brand and citizens.

Nowadays the Web represents a medium through which businesses can effectively disseminate and demonstrate their commitment to manage the impacts of their operations, to prove their accountability and integrity.

Sharing information on the Web offers significant reach and universal exposure to Web users, allowing users simplicity of access at any time and from anywhere. New data can be consulted in a timely fashion as soon as available. Given the enormous potential offered by this channel, being sustainable, and communicating it, is a winning strategy for businesses.

A third assumption relates to the form in which the sustainability information is provided.

Companies can share these practices on the Web in several ways, such as a report in the form of a PDF attachment or dedicated HTML page (or pages) or both, sometimes is a video from the CEO or other times is an entirely separate website devoted to the thematic.

For the purpose of this initial investigation, any sustainability information published on the company website in the sole form of HTML content is considered.

It will be left to future work to revise these assumptions and to provide a more detailed definition of what content is best suited for the purpose of the indicator, for example whether to also include formal reports in the form of PDF files or not.

2 The web scraping application

Web scraping is a technique employed to extract data from websites programmatically, without the need of a user interaction. To know which companies publish sustainability content on their websites, we could search for each company on the Web, inspect each website manually and note down if there is any sustainability content available. Instead of manually inspecting websites, the web scraping program can interact with websites like a web browser does. Rather than presenting the data served by the website on a computer screen, the web scraping program can extract and eventually save the required data to a local file or database, performing our same task within a fraction of the time.

To begin, any web scraping project requires a list of websites addresses or starting URLs[3] for the scraper to visit. In this case, a list of companies' websites, preferably homepages. How these URLs were acquired is described in Section 2.1.

The web scraping program will access all the URLs provided and look for pages that might contain sustainability information.

Once a page is found, it needs to be cleaned and standardised in a suitable format before it is saved in the database.

A detailed description of the web scraping program is given in Section 2.2, while Section 2.3 gives a more in-depth overview of the two last step of the web scraping process: the extraction of the main content from the web pages and the connection to the database.

2.1 The starting URLs

For the purpose of this project, the following sources of companies data were considered:

- 631 companies listed on the London Stock Exchange FTSE All-Share[4]
- an extract of the UK's largest [5] businesses from the ONS Inter-Departmental Business Register (IDBR)[6]
- OpenCorporates [7] extract of ~13,000 companies with a website information

---

[3] URL stands for Uniform Resource Locator. URL is the global address of documents and other resources on the World Wide Web. The term "Web address" is often used as a synonym for URL.

[4] The constituents of the FTSE All-Share Index total 631 companies:
`http://www.londonstockexchange.com/exchange/prices-and-markets/stocks/indices/summary/summary-indices-constituents.html?index=ASX`
Each company has a profile page with information on Company address, Company website, Market cap, Listing/Admission to trading, Trading service, Market and Listing category.

[5] The data was extracted by selecting businesses with at least 250 employees to comply with Eurostat definition of large enterprises:
`http://ec.europa.eu/eurostat/statistics-explained/index.php/Glossary:Enterprise_size`

[6] The IDBR is the ONS Inter-Departmental Business Register which is is a comprehensive list of UK businesses used by government for statistical purposes. The 2 main sources of input are the Value Added Tax (VAT) and Pay As You Earn (PAYE).

[7] OpenCorporates is an Open Database of corporates around the World. For the UK, OpenCorporates held data for almost 4million active companies. For each company there is a profile page. However, only a small subset of these companies also have a website information: `https://opencorporates.com/companies/gb?current_status=Active&q=&types_of_data_held=Website`

- the *Top Track 100 league table 2016*[8] published by the Times

At last, the Top Track 100 league table 2016, a list of the largest 100 private companies in the UK ranked according to sales, was chosen on the basis of these considerations:

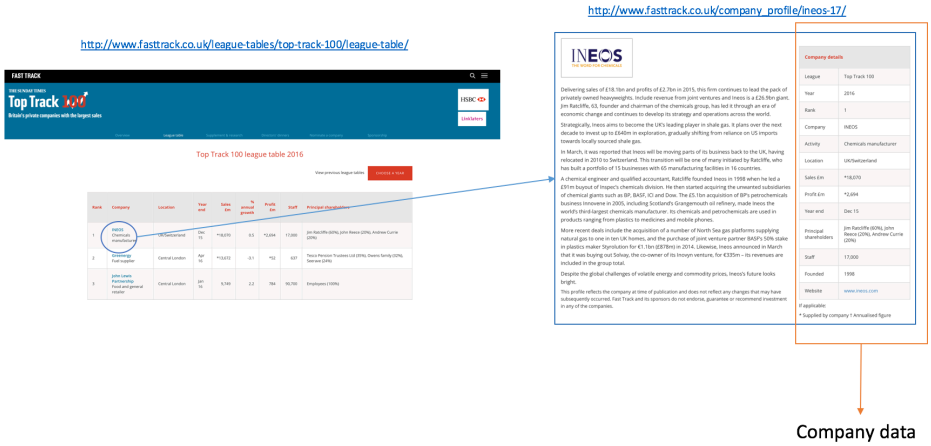
- **Size of the list:** 100 companies is a manageable number for a proof-of-concept.
- **Comparability of the data:** it is a list of the largest 100 private businesses in the UK, ranked according to sales. It suits the definition of the indicator which suggests a focus on large and transnational companies. This reason led to discard the OpenCorporates source, as it does not provide any information on the size of the businesses.
- **Unrestricted use:** it is public data. Sensitivity around use was one of the reasons for discarding IDBR for the purpose of this project. However, for the production of an official indicator, the IDBR should be considered as the main source.
- **UK-based companies:** all 100 companies are based in the UK (conversely, a large slice of the Companies on the Stock Exchange are international companies, who chose to be listed in the UK).
- **Additional data:** the table reports the latest economic characteristics such as profit, sales, the number of employees and industry. This information was not available for any other source or marked as sensitive information in the case of the IDBR.
- **Available websites:** the table contains an up-to-date website address for each company, fundamental to build the list of starting URLs for the scraper. The website information was also public on London Stock Exchange and accessible via API for the OpenCorporates data, but not available in the IDBR.

Although the full list of details of each single company is not immediately visible on the front table, it can be accessed by following the link under the company name. The link leads to each specific company's full profile, (for example the INEOS profile page[9]).

[8] The Sunday Times HSBC Top Track 100 league table ranks Britain's 100 private companies with the biggest sales. It is compiled by Fast Track and published in The Sunday Times each July http://www.fasttrack.co.uk/league-tables/top-track-100/league-table/

[9] http://www.fasttrack.co.uk/company_profile/ineos-17/

Figure 1: A screenshot of the 2016 Top Track 100 league table (on the left) and the INEOS specific company's profile page (on the right).



A script was created to:

1. Extract all links that lead to the companies' profile pages
2. Access all 100 companies' profile pages and extract the company data, wrapping them in a table.

2.1.1 Step 1: Get URLs that lead to the companies' profile pages

The first step is to extract all the URLs that hide in the HTML code behind the companies' names. These URLs will then be used to acquire each company's specific data in Step 2. In order to do that, the script accesses the Top Track 100 league table 2016 page using the Python requests[10] library. A HTML parsing library, BeautifulSoup[11] , is then used to parse the HTML content and retrieve all the <a> Tags (<a> is the HTML tag which indicates there is a link) that contain a URL to the company profile page.

Figure 2: First three URLs extracted.

```
http://www.fasttrack.co.uk/company_profile/ineos-17/
http://www.fasttrack.co.uk/company_profile/greenergy-12/
http://www.fasttrack.co.uk/company_profile/john-lewis-partnership-15/
```

2.1.2 Step 2: Get the details for each company

Now that we have all the URLs that lead to the company profile pages, the script can access those links and extract the detailed company data.

The script accesses the company URLs extracted in Step 1 one by one, and grab the Rank, Location, Sales £m, Profit £m, Year end, Founded, Website and Staff information. Results are saved in a table.

---

[10] Requests allows you to HTTP/1.1 requests, without the need for manual labour, and have back the HTML page: `http://docs.python-requests.org/en/master/`

[11] BeautifulSoup is a popular Python library to parse HTML pages, to easily navigate their tree structure and extract data from them: `https://www.crummy.com/software/BeautifulSoup/`

Table 1: First five rows of the data extracted

Rank	Company	Location	Sales £m	Profit £m	Year end	Founded	Website	Staff
1	INEOS	UK/Switzerland	18,070	2,694	Dec-15	1998	http://www.ineos.com	17,000
2	Greenergy	Central London	13,672	52	Apr-16	1992	http://www.greenergy.com	637
3	John Lewis Partnership	Central London	9,749	784	Jan-16	1864	http://www.johnlewispartnership.co.uk	90,700
4	Swire	Central London	7,238	1,229	Dec-15	1816	http://www.swire.com	81,833
5	Palmer and Harvey	East Sussex	4466	35	Apr-15	1925	http://www.palmerharvey.co.uk	4,330

The Website and Company name are extrapolated to create the seed.txt file. The seeds.txt file is the input file of the web scraping program.

Figure 3: First three rows of the seeds.txt.

http://www.ineos.com, INEOS
http://www.greenergy.com, Greenergy
http://www.johnlewispartnership.co.uk, John Lewis Partnership
http://www.swire.com, Swire
http://www.palmerharvey.co.uk, Palmer and Harvey

2.2 The scraper

Granted the seeds URLs, this section introduces the scraper. In simple words, what the scraper does is:

For each company website:

1. Extract all the links and the text on top of those links
2. If the link leads to a page that might contain sustainability information:
    a. Open the page Extract the main textual content (excluding menus, navigation bars, footers, etc.)
    b. Save it to the database.

A search by keywords can be used in Step 2 to flag the link that suggests the presence of a sustainability page. The scraper specifically looks if a keyword is present in the URL of the link or on the text on top of the URL that leads to the sustainability content. Keywords were chosen by inspecting a sample of websites and looking at what were the most common words used to introduce sustainability content.

The list of keywords is as follows:

- csr
- environment
- sustainab
- responsib
- footprint

"csr" stands for Corporate Social Responsibility.

"sustainab" and "responsib" are both stemmed[12] in order to match word variations such as "sustainable" and "sustainability".

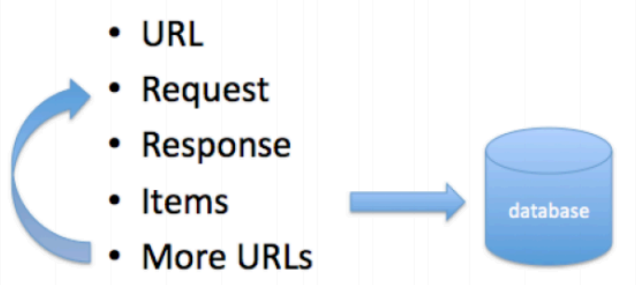
The scraper will need to navigate through the website, accessing every internal link. This type of action is defined as crawling. The general approach to an exhaustive website crawl is to start with a top-level page (such as the homepage) and search for a list of all internal links on that page. Every one of those links is then crawled, and as additional lists of links are found on each one of them, triggering another round of crawling. While recursively traversing each website, the scraper flags only the pages that suggest sustainability content, i.e. at least one of the predefined keywords was found in the URL of the page or the text of the hyperlink leading to the page. Crawling an entire site, especially a large one, is a memory intensive process that is best suited to applications where a database to store crawling results is readily available. To this end, once a page is flagged, it is processed to keep only the main content of it and then sent to the MongoDB[13] database.

A description of the architecture of Scrapy[14], the framework used to develop the scraper, and a detailed explanation of the scraper flow follow in the next sections.

2.2.1 Scrapy and the web scraping process

Scrapy is an application framework, written in Python, for writing web scrapers that scrape websites and extract data from them. Scrapy is a powerful tool that handles many problems associated with web scraping. It handles much of the complexity of finding and evaluating links on a website, crawling domains or lists of domains with ease, makes sure all URLs are unique, normalises relative URLs where needed; and recurses to go more deeply into pages. Scrapy also takes care of a lot of the lower level async programming, which is required to get good performance. The framework helps to organise the structure of the scraper around a series of logical components. These components are essential parts of any web scraping process and in Scrapy they come together in a sequence of recursive steps that goes under the name of UR²IM process [4].

Figure 4: The Basic Scraping Equation – UR²IM.
Source: Kouzis-Loukas, D. (2016). Learning Scrapy.



[12] The stem is the root or main part of a word, to which inflections or formative elements are added.

[13] Free and open-source, cross-platform, NoSQL and document-oriented database: `https://www.mongodb.com/`

[14] An open source Python framework for extracting the data from websites: `http://scrapy.org/`

2.2.1.1 URL

At the beginning of the process, the scraper picks one of the seeds, one of the starting URLs, which dictates the domain[15] boundaries for the following steps. The scraper repeats the process for every URL in the seeds.txt file.

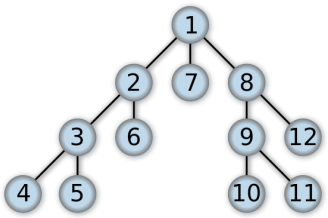
2.2.1.2 Request objects

A Request object represents an HTTP request, which is usually fired by the scraper, executed and returns a Response object.

When the scraper accesses the company homepage, it will fire as many requests as internal links present on that page. Every one of those links becomes a new request, and as additional links are found on each one of them, an additional round of requests will be triggered. If the URL link or the text on top of the URL contains any of the predefined keywords the Request is flagged and this information is added to the metadata of the Request.

By default, Scrapy crawls in Depth-first order (DFO)[16]. DFO is an algorithm for traversing or searching tree or graph data structures (most websites assume the form of a tree structure).

Figure 5: The order in which pages of a website are visited in DFO order. Source: https://en.wikipedia.org/wiki/Depth-first_search



It starts at the tree root and explores as far as possible along each branch before backtracking.

Sometimes this approach can lead the scraper to get stuck at one branch of the tree. For example, if page 2 is a 'Products' page, it can contain hundreds of links, and each of these links can contain just as many links, making the time to access all the lower levels of page 2 to increase exponentially.

To improve the efficiency of the scraper and give priority to high-level pages, a BFO (Breadth-first order)[17] approach is imposed to the scraper.

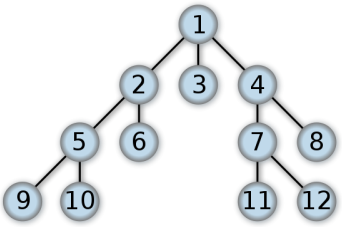
---

[15] One of the most important part of an URL is the domain name. In the URL http://www.example.com/index.html, the domain name is example.com. The domain name tell us whether an URL is an internal link, i.e. it is a page of the same website and thus must share the same domain name, or it is an external link, i.e. it links to another website and thus it has a different domain name.

[16] See `https://en.wikipedia.org/wiki/Depth-first_search`

[17] See `https://en.wikipedia.org/wiki/Breadth-first_search`

Figure 6: The order in which pages of a website are visited in BFO order. Source: https://en.wikipedia.org/wiki/Breadth-first_search



It starts at the tree root and explores the neighbour nodes first, before moving to the next level neighbours.

In our web scraping scenario, the starting node (1) would be the company's homepage, the starting URL, and the set of first level linked nodes (2, 3, 4) would be the other pages that are immediately accessible from the homepage. Most environmental and sustainability information is usually linked from the homepage of the corporate website, i.e. can be immediately accessed in the first depth level. By imposing the BFO order, the scraper will execute the requests following the depth level order. If the scraper does not find any sustainability information in the first round of requests, it is likely that there is no sustainability page on the website. Potentially, this can translate in a future stopping rule so that the scraper can avoid wasting time on the website and move to the next one.

2.2.1.3 Response objects

A Response object represents a fetched HTML page. Instead of being displayed in a web browser, this page is passed directly to the scraper for processing. The Response object keeps the same metadata of the Request who generated it, so the information whether the page has been flagged or not is known. If there were keywords, then an Item is generated. If not, the scraper again extracts all internal links from the web page, generating new Requests.

2.2.1.4 Items and Item Fields

The main goal in web scraping is to extract structured data from web pages, which are unstructured sources.

To facilitate the collection of the scraped data, Scrapy provides the Item class. Items are simple containers that provide a dictionary-like API, with a convenient syntax for declaring their available fields. The Field class is an alias to the Python built-in dictionary class (key-value pairs).

When a sustainability page is found, a new Item is generated with the following fields:

- Company name
- Link URL
- Link Text
- List of Keywords that matched Full HTML

2.2.1.5 Item Pipeline

After an Item is created with the data extracted from a Response object, it is sent to the Item Pipeline. The Item Pipeline consists of several components, and the Item goes through each one of them sequentially, according to a predefined order.

Each pipeline component is a Python class that implements a simple method: it receives an Item and performs an action over it. A component can also decide if the Item should continue through the pipeline or be dropped and no longer be processed.

- Item pipelines are typically used for:
- cleansing HTML data
- validating scraped data (checking that the items contain certain fields)
- checking for duplicates (and dropping them)
- storing the scraped item in a database

In this project, the Item Pipeline has two components: the class that deals with the extraction of the main content from the "Full HTML" and the class that submit the Item to the MongoDB database. Section 2.3 describes in greater details these two important pieces of the process.

2.2.1.6 More URLs → Link Extractors

Link extractors are objects which extract links from web pages which will be eventually followed. Link extractors facilitate the extraction of those links which have the same domain as the starting URL, i.e. only the internal links of a website. A URL extracted by Link extractors becomes a new Request only if the URL has not been seen before, to avoid to generate duplicate Requests.

2.3 The Item pipeline: Extracting the text and saving to the database

Before it is sent to the database, the Item goes through the Item pipeline, here implemented as a sequence of two steps executed in order. In the first step, the *ContentExtractor*, the field "Full HTML" is replaced with a new field "content", which differ from the previous one because it contains just the main text visible on the web page. In the second step, the *MongoConnector*, the modified Item is validated and finally sent to the MongoDB database.

What these two components are, it is described in the following two sub-sections.

2.3.1 The ContentExtractor class and Content Extraction algorithms

Web pages are often cluttered with additional features around the main content. These "features" frequently include elements such as navigation panels, pop-up ads, advertisements and comments. These noisy parts tend to affect the performances of NLP tasks negatively.

Over the years, content extraction algorithms have been developed to analyse a web page structure and separate the main article content from the surrounding clutter. The removal of these noisy elements is non-trivial because HTML does not enforce structural constraints on the way an HTML page should be composed. HTML is not a semantically strict language, and thus content providers are free to compose web pages in whatever way they like. Although the latest standard, namely HTML5, offers new semantic elements to define different parts of

a web page such as <article>, <nav> and <footer>, the standard has not been widely adopted.

A comparison of four different content extraction methods was performed on a sample of 30 URLs to decide the most suitable for this particular task.

At first, the content corresponds to a full HTML page, with tags, scripts, styles, and so on. From the full HTML, the main text was extracted using the following four methods:

1. Readability

Readability was created to be used as a browser add-on (a bookmarklet). With one click it transforms web pages for easy reading and strips away clutter. Readability was originally thought to turn any web page into a clean view, to give the user a better reading experience.

Given its success among users, the original Readability codebase has been embedded in several applications, including Apple's Safari 5 browser, the Amazon Kindle and popular iPad applications like Flipboard and Reeder.

The library used here, python-readability[18], is the Python port of a ruby port of the original arc90's Readability project[19].

The algorithm gives a score to each part of the HTML page based on a series of deterministic rules such as the number of commas, class names, link density, and various others. For example, points are given based on keywords extracted from the HTML-ID-names and HTML-CLASS-names of the HTML Tags surrounding the text. At first, two lists of words are defined: one list contains IDs and CLASSes with a positive meaning (e.g. post, content, text), the other list contains IDs and CLASSes with a negative meaning (e.g. comment, meta, footer). If a tag has a positive ID or CLASS, it will get additional points; if it has a negative ID or CLASS, it will lose points. Another measure is the link density of the text. Good content should normally have a relatively small link density (5% or less).

2. Dragnet

Dragnet [5] uses machine learning models to extract the main article content and optionally user-generated comments from a web page. It is the result of an ensemble of diverse feature sets and algorithms.

The Dragnet Python library[20] allows to apply the method directly to the web page and quickly get the content extracted as a result. The machine learning models are already trained on a set of real web pages.

However, it is also possible to retrain the models and optimise them on a set of new pages, with own-defined training and test sets. The latter option could be worth exploring for future developments. For the time being, only the pre-trained option was considered.

[18] `https://github.com/buriy/python-readability`
[19] `http://lab.arc90.com/experiments/readability/`
[20] `https://github.com/seomoz/dragnet`

3. **BeautifulSoup get_text()**

BeautifulSoup[21] is a Python library for extracting data out of HTML files. The get_text() method of this library returns all the text of an HTML document. It is the most inclusive method compared to the others because it returns the clutter together with the informative content without trying to make any distinction. This method can be considered as the base model, against which to assess the performance of the other methods.

4. **<p> Tags**

This method simply extracts all <p> Tags from the web page. This method is the most restrictive compared to the other methods. It limits the extraction to the solely <p> Tags and therefore excludes titles tags, <h1> <h2> ... <h6>, and other tags that are often used to display text: <li>, , and so on.

2.3.1.1 Benchmarking

The four extraction algorithms mentioned above were benchmarked against the true content, the gold truth, manually extracted for each of the 30 URLs by a simple action of copying and pasting.

The comparison was useful for two reasons:

- the best method performing on the test was integrated into the scraper, in the *ContentExtractor* step of the Item pipeline, to extract the main textual content from web pages
- overall the comparison gave an early indication on how well the scraper is performing on a sample of 30 websites.

Three types of comparisons were considered to evaluate how methods perform against the true content:

1. Comparison by number of words
2. Comparison by content similarity computed using the divergence distance between the words distributions: how distant is each of the four texts from the true content?
3. Precision, recall and the F1 score, which, as explained later, can be used to quantify each method performance.

1. **Comparison by number of words**

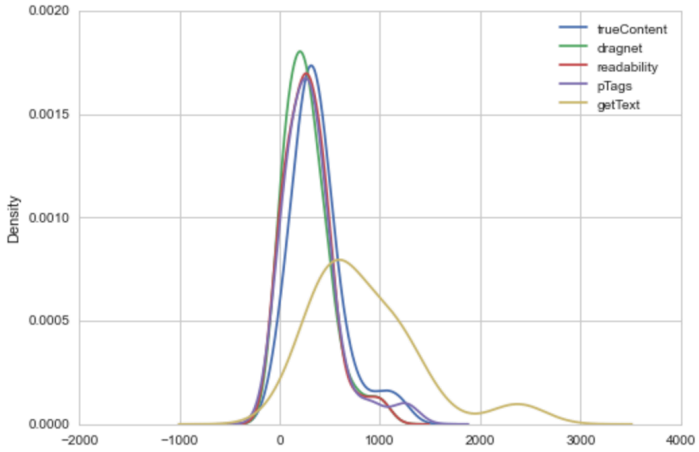
The result is a table of counts with five columns (the true content, dragnet, readability, <p> Tags and get_text()) and N rows, where N corresponds to the number of URLs in the sample.

[21] `https://www.crummy.com/software/BeautifulSoup/`

Table 2: An extract of the table of words counts of each method applied to first five web pages.

Doc N.	True content	Dragnet	Readability	<p> Tags	get_text()
1	22	14	43	40	313
2	617	211	211	509	780
3	473	467	470	247	1157
4	384	180	362	368	1105
5	363	360	371	390	1068

Figure 7: A visual inspection of the distributions of counts



It is clear that the get_text() method extracts often more words than it should. The get_text() does not try to make any discrimination between the clutter and the true content, it just takes both, and thus a distribution of higher words counts was expected.

The others methods are closer to the true content, as they follow the true content counts distribution quite well. However, this words counts comparison does not make any distinction between the words themselves. Thus, the lack of some words can be balanced out by the presence of clutter words.

2. Comparison by content similarity

The extracted contents are considered here as vectors. This type of representation is called bag-of-words[22]. Each text is represented by one vector where each vector element consists of a sequence of (word_id, word_weight) tuples. The word weight is simply the normalised word count, thus representing the contents as distributions over words.

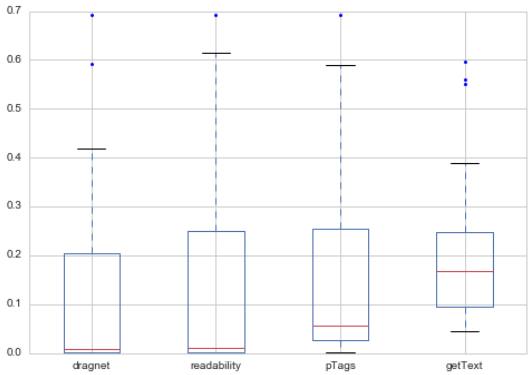
For each URL, the Jensen–Shannon divergence[23] (lower is better) is used to quantify the distance of the contents extracted by each content extraction method from the true content.

[22] See Section 4 for a more detailed explanation.
[23] See Section 5 for a more detailed explanation.

Table 3: Distances from the true content for the first five documents

Doc N.	Dragnet	Readability	<p> Tags	get_text()
1	0.69	0.69	0.69	0.55
2	0.19	0.19	0.06	0.04
3	0.00	0.00	0.17	0.22
4	0.14	0.01	0.03	0.25
5	0.00	0.00	0.02	0.25

Figure 8: Boxplots of the distances from the true content



The boxplots in Figure 8 shows that the methods dragnet and readability have most of the distances close to 0 (the red line indicates the median). The dragnet and readability boxplots suggest that the text extracted by these two methods is most of the times close to the true content (many documents have distance close to 0), with some exceptions.

This second comparison highlights the gap between the dragnet-readability and <p>Tags-get_text() methods.

3. Precision, Recall and F1 score

The following table compares the token-level performance.

For each web page and each of the four methods:

- precision: it is the percentage of true tokens among all the retrieved tokens by the method

$$precision = \frac{|\{actual\ tokens\} \bigcap \{retrieved\ tokens\}|}{|\{retrieved\ tokens\}|}$$

- recall: it is the percentage of true tokens retrieved by the method among all the true tokens

$$recall = \frac{|\{actual\ tokens\} \bigcap \{retrieved\ tokens\}|}{|\{actual\ tokens\}|}$$

- F1 score: it is the harmonic mean of precision and recall

$$F1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

where $|\{actual\ tokens\} \cap \{retrieved\ tokens\}|$ is the number of tokens (words) captured by the content extraction algorithm that match the tokens of the true

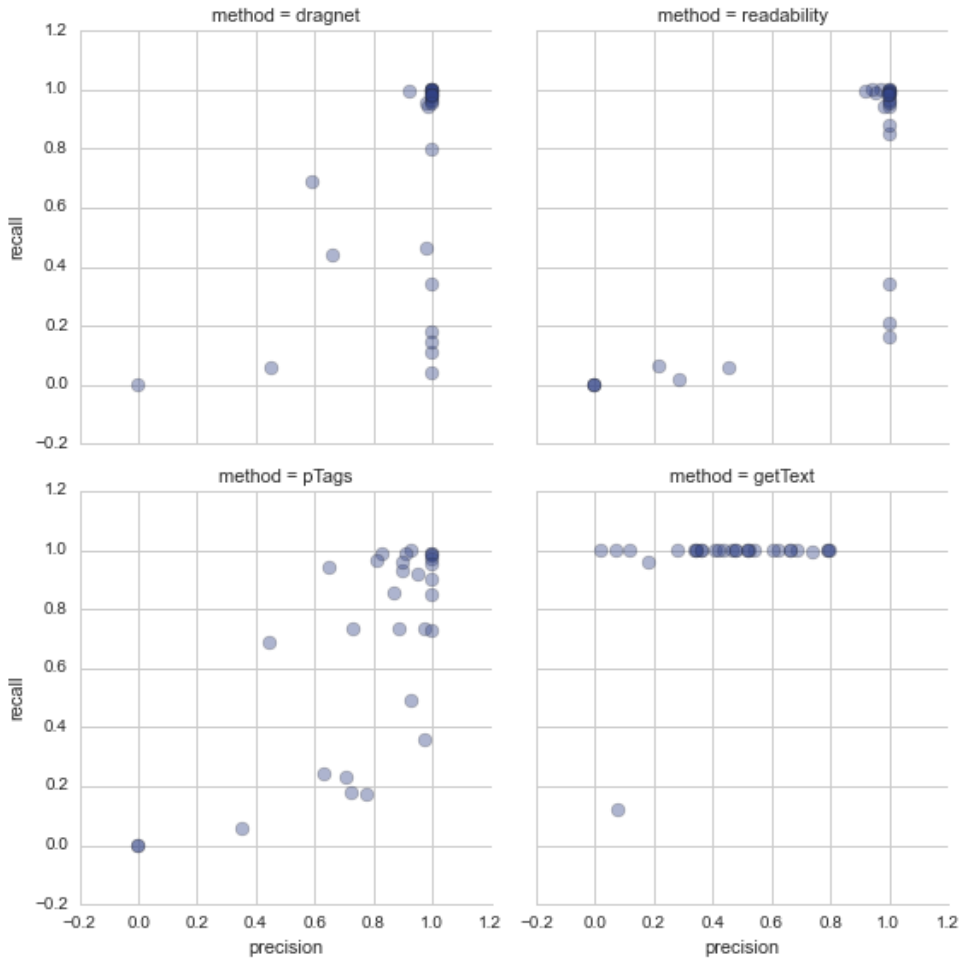
content, $|\{retrieved\ tokens\}|$ is the number of tokens retrieved by the content extraction algorithm and $|\{actual\ tokens\}|$ is the number of tokens of the true content.

Precision can be seen as a measure of exactness or quality, whereas **recall** is a measure of completeness or quantity.

- Here, a high precision indicates that on average most of the retrieved tokens are relevant to the true content, i.e. there are not many false positives. Instead, the get_text() method, which retrieves all text from the web page, obviously has the effect to produce many false positives and downgrade the quality of the retrieved content.
- On the other hand, the recall indicates how many of the actual tokens the method is able to retrieve. The dragnet, readability and the <p> Tags methods have a lower average recall with respect to the baseline. Low recall indicates that the first three methods tend to retrieve less content. When get_text() method retrieves all text instead, with no surprise, the higher recall shows that it can catch almost all the true content.

Precision, recall and the F1 score are calculated for each page and each of the four methods.

Figure 9: Precision vs. Recall. Each point represents a website.



Contents extracted with the dragnet and readability achieve most of the times full precision. The recall varies between 0 and 1. In the case of the <p> Tags method, precision also tends to vary, although it is most of the times above 0.5. The get_text() method tells a different story: precision varies widely between 0 and 0.8, whereas the recall is pretty constant to 1.

Finally, the scores are averaged among all documents and Table X shows the average scores for each method:

Table 4: Average precision, recall and F1 score for each method

	Dragnet	*Readability*	*<p> Tags*	*get_text()*
Precision	0.92	0.82	0.80	0.46
Recall	0.73	0.71	0.68	0.97
F1	0.76	0.73	0.71	0.59

For the purpose of our NLP task, it is more important to have a high precision, i.e. a high quality of the retrieved content, compared to recall and the F1 score.

The lower recall implies that the dragnet method will probably retrieve less content. On the other side, there is the guarantee that the extracted content is relevant for the NLP task.

2.3.1.2 Test results

The get_text() method performed worst in all the comparisons, as it was expected. It did achieve a high recall score in the third comparison, meaning it can grab the whole content of the page nearly all the times. However, this was not sufficient given that for the purpose of the NLP task a high precision is required to get an accurate performance.

The dragnet and readability libraries standout in all the comparisons. They make use of sophisticated algorithms which allow them to distinguish themselves from the simpler methods. The high precision gives confidence on the quality of the extracted content.

Lastly, the performance of the <p> Tags method is considered in between. The suspicion is that the performance of this method is highly sensible to the quality of the website. In a page built following HTML good practices, the <p> Tags should enclose all blocks of text, i.e. paragraphs. This is not always the case, and varies quite widely across the industry, meaning that there is no consistency in the results.

As per the overall performance of the scraper, the result of the get_text() method is promising: the content is there but need to be extracted in a better way. When the text is returned by the get_text() method, as the word suggest, it is just plain text. So, at this point, it is not possible to separate the good from the bad content. On the other side, neither the dragnet, nor the readability, nor the <p> Tags, can achieve the same level of performance in terms of recall.

Two options to pursue in future could be:

1. try to intervene before the HTML tags are discarded by the get_text() method so that it might still be possible to make the distinction
2. proceed with the option offered by the dragnet library to retrain the machine learning models and optimise them on a set of own-defined pages.

It is important to acknowledge that this test covered 30 randomly selected URLs. Higher confidence on the method and performance of the scraper could be achieved by considering multiple samples.

On the light of the three types of comparisons made during the test, the dragnet library is the one which is considered to have performed better. When the *ContentExtractor* class of the Item Pipeline receives the Item with the full HTML, it extracts the main content using the dragnet library. The new item will have the full HTML field replaced by the content extracted. The Item then proceeds to the next component of the Item pipeline, which checks the validity of the Item before sending it to the MongoDB database.

2.3.2 The MongoConnector class and the MongoDB database

MongoDB is a cross-platform, document-oriented database that provides high performance, high availability, and easy scalability. The two important structures that form the basis of the MongoDB architecture are **collections** and **documents**:

- A collection is a group of MongoDB documents. A collection is often considered the equivalent of the table structure in the relational database.
- In MongoDB, documents are JSON objects, i.e. a set of key-value pairs.

The MongoDB database was selected to store the items generated by the scraper. The choice was made on the basis of the following considerations:

- **The dynamic schema of MongoDB documents:** documents within the same collection can have different fields or structures, and common fields in a collection's documents may hold different types of data.
- **MongoDB documents support of several data types:** documents can contain many different key-value pairs, or key-array pairs, or even nested documents. Values can be of type number, string, boolean as well as array and hash.
- **Explicitly declared Indexes:** Indexes support the efficient resolution of queries. Without indexes, MongoDB must scan every document of a collection to select those documents that match the query statement. This scan is highly inefficient. MongoDB allows defining indexes on any field or subfield of the document in a collection. Compound indexes are user-defined indexes on multiple fields.

- **Easy integration with PyMongo:** PyMongo [24] is a Python distribution containing tools for working with MongoDB. PyMongo allows to connect the scraper to the database, so whenever a new item is created out of a web page, can be immediately stored.

The *MongoConnector* class is responsible to communicate with the database. However, before it is sent out, each Item is validated by checking that at least the company (name) and URL fields exist. The company (name) and URL fields ensure a unique compound index for the collection, and thus are required to be valid and not null.

Table 5: Fields, data types, and description of the MongoDB documents that will store the scraped web pages

Field	*Data type*	*Description*
_id	ObjectId	Unique Index automatically created by MongoDB for each document
company	string	Company Name. This field is part of the compound key for this collection
url	string	The URL of the webpage. This field is part of the compound key for this collection
link_text	string	The link text, i.e. the text that was used to indicate the URL on the page where the link was found
content	array	The textual content of the webpage extracted using the dragnet library.
keywords	array	The list of unique keywords found either in the link text or in the link URL

Figure 10: An example of document

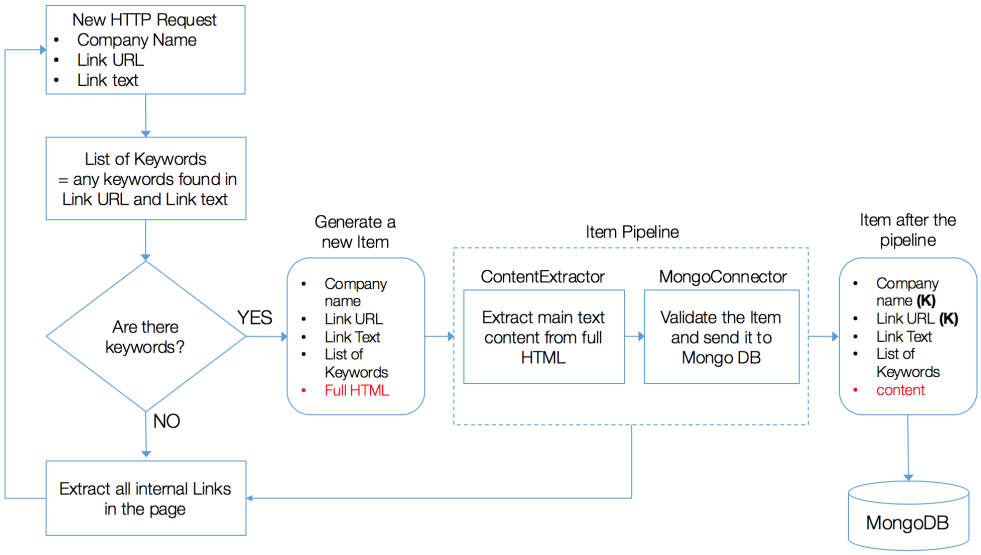
```
{
    _id:        ObjectId(7df78ad8902c)
    company:    "John Lewis Partnership",
    url:        "http://www.johnlewispartnership.co.uk/content/cws/csr/
                our-environment/operational-emissions/our-buildings-and-site-emissions.ht
                ml",
    link_text:  "Operational Emissions",
    content:    "We remain committed to increasing the energy efficiency of our
                buildings, procuring low carbon energy and finding more efficient ways to
                distribute our goods, as well as encouraging innovation
                Our overarching aim is to achieve a 65% reduction in carbon intensity
                (tonnes per £m sales) by 2020/21 against a 2010 baseline. We have also
                developed a number of targets which focus on improving energy
                \nperformance, refrigeration management and transport operations. This is
                in order to demonstrate our continuing commitment to resource reduction
                alongside low carbon procurement.
                [...] ",
    keywords:   [ "environment", "csr" ]
}
```

[24] `https://api.mongodb.com/python/current/`

2.4 Wrapping up: The scraper flow

Combining all the pieces together, the diagram in Figure 11 shows a high-level overview of the scraper flow.

Figure 11: An high-level overview of the scraper flow



The **(K)** after Company name and Link URL indicates that these two elements are Indexes.

"Full HTML" and "content" are highlighted in red to emphasise the fact that the "Full HTML" field is replaced by the "content" extracted.

3 Statistics of websites collection

This section provides a general overview of the work done by the scraper.

3.1 Number of web pages scraped

The scraper extracted a total of 563 webpages.

3.2 Number of companies for which at least a page was scraped

The 563 pages were collected from 59 companies.

When checking why there was no presence of the others 41 companies, the following 6 problematic cases were identified:

- 2 websites had an entrance form: one requiring the location, to load the website specific for the region, and the other requiring the age. Forms require special treatment, but for the time being the scraper was not able to scrape them. Both websites had sustainability content.
- 1 website had the sustainability content published on a separate website: since the scraper does not follow external links the content was missed
- 1 website was not working for several days. Sometimes it happens that websites are down for maintenance. However, in this case, despite retrying several times, the website was still not accessible
- 1 website had some content related to sustainability in the "Mission & Values" page
- 1 website had only content related to a charity foundation. No keyword was matched and so the content was missed.

The remaining 35 companies did not have any sustainability pages published on their websites.

Overall: 59 companies were successfully identified, the scraper couldn't access or did not find the sustainability content for 6 companies, and 35 companies did not have any sustainability content published.

Two additional findings were found:

1. Excluding the 6 companies for which the scraper had problems, if we consider the companies ranked according to sales, in the first half[25] of the rank, 80% of the companies have sustainability-related content on their websites. In the second half of the rank, only 45% of the companies have sustainability-related content on their websites.
2. Companies on top of the rank are mainly construction or manufacturing companies, and big retailers whereas companies in the bottom are mostly companies working in the service industry (recruitment

[25] Excluding the 6 companies for which the scraper had problems, 94 companies are left. The remaining companies are divided in two halves of 47 companies each.

consultancies, travel agencies and so on). Thus it appears the industry sector might play a significant role in whether the company publish the sustainability content or not. However, this is not always the case because some service companies in the list do still publish sustainability content.

The extent to which the numbers of sustainability reporting are affected by the industry and size of the company are worth exploring more, especially when a much larger number of companies will be considered.

3.3 Keywords rank

Table 6: Number of times each single keyword appears among the whole collection, ordered from the most to the least used

Keyword	*Count*
responsib	234
environment	171
sustainab	169
csr	92
footprint	25
policy	10

Additional terms that could be considered for future inclusion could be: ethical, charities, foundation.

3.4 Keywords combination rank

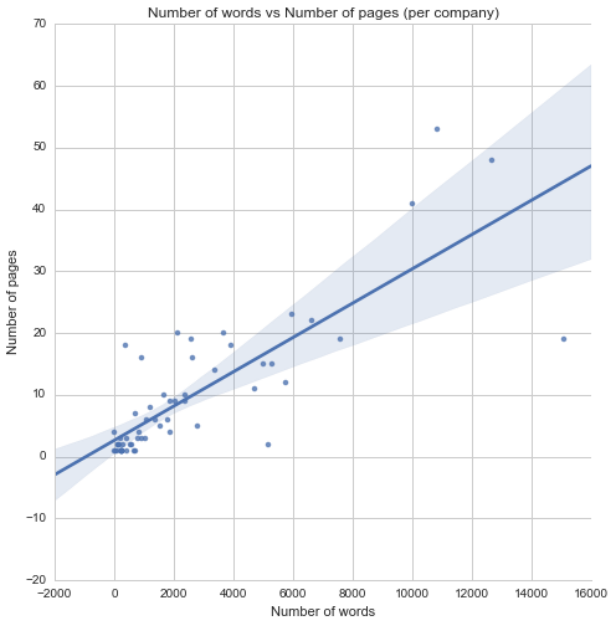
Table 7: Number of times each combination of keywords appears among the whole collection, ordered from the most to the least used

Keywords combination	*Count*
sustainab	141
responsib	139
environment	75
csr	61
environment, responsib	60
footprint	18
csr, responsib	17
environment, sustainab	11
csr, environment	10
responsib, sustainab	7
environment, footprint	6
csr, responsib, sustainab	4
environment, policy	4
environment, responsib, sustainab	3
policy, responsib	3
policy, sustainab	2

environment, footprint, responsib	1
environment, policy, sustainab	1

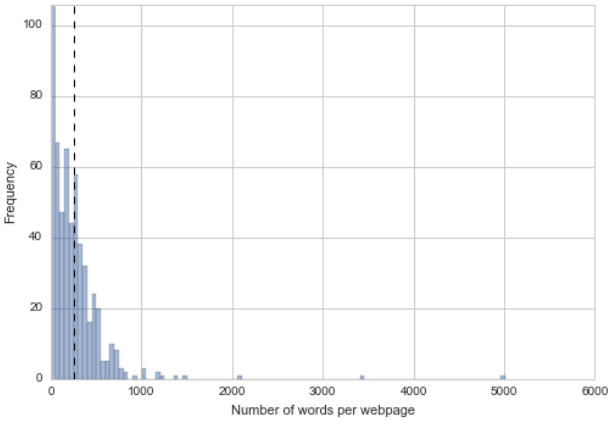
3.5 Number of words, Number of pages

Figure 12: Number of words scraped per company vs. Number of web pages per company



Before any cleaning and preprocessing of the text is done, Figure 13 shows the distribution of the number of words per scraped web page. On average, the web pages scraped have ~260 words. The distribution is skewed to the right, with few pages having more than 1,000 words.

Figure 13: Distribution of the number of words per web page



4 Text Preprocessing

After collecting the data from the Web, the preprocessing of the textual content extracted is the next step.

Preprocessing the text is an important task and critical step to be performed before any actual analysis.

Two main reasons justify the need of text preprocessing:

1. To reduce file size of the text documents
2. To improve the efficiency and effectiveness by matching the similar words in a text document or removing words which do not add much to the content.

Following the usual pipeline of dealing with text documents, a number of text preprocessing and text cleaning steps were considered. They include tokenization, normalising case, removing user-defined or common stopwords, and word stemming.

4.1 Normalising text

4.1.1 Case-folding

A common task of reducing all letters to lowercase. Case-folding is used to "normalise" text for the purposes of comparison.

4.1.2 Tokenization

Tokenization is the process of converting a document to its atomic elements. What are the atomic elements of a text does not officially have a standard definition, but usually, depends on the data and the tasks to be performed later.

Tokenization can break a stream of text up into words, phrases, sentences or symbols. Typically, tokenization occurs at the word level, and the result is a list of tokens which does not include punctuation and whitespaces.

4.1.3 Stopwords

Stopwords refer to words with a lack of content that do not contribute to the context or content of textual documents. These words usually are articles such as “a” and “the” or pronouns such as “that” and “my”.

The SMART stopwords[26] list is a set of 571 English stopwords from the SMART[27] (System for the Mechanical Analysis and Retrieval of Text) information retrieval system, developed at Cornell University in the 1960s. SMART stopwords extend the standard English set of stopwords by including forms such as “I have” and “I will” and a series of other words, such as “willing” and “regarding”.

Sometimes, additional stopwords are context-dependent. In the scraped companies' webpages, for example, the company names can be considered stopwords. In fact, a group of pages extracted from the same website it is likely to

[26] http://www.lextek.com/manuals/onix/stopwords2.html

[27] https://en.wikipedia.org/wiki/SMART_Information_Retrieval_System

repeat several times the company name across the pages. An algorithm of text analysis might then pick up this as an indication of similarity. The removal of the names has the effect to anonymise pages across the different companies and make them comparable with one another.

4.1.4 Stemming

Stemming refers to the term normalisation process of reducing derivationally related words with similar meanings to a single invariant root form. The root form is called the stem.

According to a recent survey [6] of stemming algorithms used in information retrieval, a stemming algorithm, or stemmer, serves three main purposes, two of which are directly relevant to our task:

1. To cluster words according to their topic. Many words derive from the same stem. For example, the stem of “drive”, “driven”, “driver” is "drive", and it is clear that they belong to the same concept.
2. Considering multiple words that share the same stem as a single one, translates into a reduction of the vocabulary to be taken into account in the process. A smaller vocabulary means less space needed to store the structures and a lower number of unique words that need to be tracked, speeding up a variety of computational operations.

The three major stemming algorithms are Porter, Snowball (Porter2), and Lancaster (Paice-Husk).

- **Porter:** the Porter stemming algorithm[28], or Porter stemmer, is a process for removing the commoner morphological and inflexional endings from words in English originally presented by Martin Porter in [7]. It is considered one of the most common stemming algorithm and also one of the gentlest stemmers.
- **Snowball:** Snowball, also referred to Porter2[29], is often regarded as an improvement over Porter from which it derives. Porter2 transforms words into stems by applying a deterministic sequence of changes to the final portion of the word. Slightly faster than the Porter stemmer, handles several special cases, not covered in the original algorithm.
- **Lancaster:** based on the Lancaster stemming algorithm [8]. The Lancaster stemmer is known to be slightly more aggressive than the PorterStemmer. It is the fastest algorithm, however often the stemmed representations of many shorter words tend to become totally obfuscated and not very intuitive.

[28] http://tartarus.org/~martin/PorterStemmer/

[29] http://snowball.tartarus.org/algorithms/english/stemmer.html

4.2 Preprocessed text

The preprocessing was done using two of the most popular libraries used when working with text data in Python:

- NLTK[30], the Python Natural Language Toolkit, and
- gensim[31], to realise unsupervised semantic modelling from plain text

The text was preprocessed applying case-folding, tokenisation based on white spaces and removing stopwords. The stemming was not applied.

After preprocessing the text, we need to store it in a format that is well suited for the later analysis. This involves the creation of the two important elements: the **Dictionary** and the **Corpus**.

- The Dictionary maps a unique integer ID to all words appearing in the texts extracted. Each unique word in the collection of web pages' contents will be combined with a numeric ID. Although this mapping step might not seem necessary at first, it is technically required because most of the text analysis algorithms rely on numerical libraries that work with vectors indexed by integers, rather than strings.
- Once the Dictionary is created out of the preprocessed collection of web pages' contents, each cleaned content can be converted to its bag-of-words representation, which models each document by counting the number of times each word appears. In this representation, each web page content is conceptually represented by one vector where each vector element represents a pair in the style of:

  (word, number of times the word appears in the text)

  Treating texts as a list of word frequencies (a vector) also makes available a vast range of mathematical tools developed for studying and manipulating vectors.

  The bag-of-words representation ignores the respective order of appearance of the words. The preprocessed web pages contents represented in this way constitutes our Corpus, ready to be used in the modelling part.

[30] NLTK is a leading platform for building Python programs to work with human language data. For more information see `http://www.nltk.org/`

[31] Software Framework for Topic Modelling with Large Corpora. For more information see: `https://radimrehurek.com/gensim/index.html`

An Example:

- Text 1: "the cat chased the mouse"
- Text 2: "the dog chased the cat"

From these two sentences, the dictionary is as follows

Dictionary: { "the": **100**, "chased": **131**, "dog": **155**, "cat": **201**, "mouse": **230** }

The dictionary is referred to count the number of times each word occurs in each sentence and to get the bags of words representation:

- V1 = [(**100**, 2), (**131**, 1), (**155**, 0), (**201**, 1), (**230**, 1)]
- V2 = [(**100**, 2), (**131**, 1), (**155**, 1), (**201**, 1), (**230**, 0)]

To save space gensim will represent each text as a sparse matrix that only records non-zero entries:

- V1 = [(**100**, 2), (**131**, 1), (**201**, 1), (**230**, 1)]
- V2 = [(**100**, 2), (**131**, 1), (**155**, 1), (**201**, 1)]

5 Topic modelling and Latent Dirichlet Allocation

Topic models represent a family of computer programs that extract topics from texts. A topic is intended here as a list of words that occur in statistically meaningful ways.

Topic modelling algorithms do not require any prior annotations or labelling of the documents. Instead, the topics emerge from the analysis of the original texts.

LDA is a special case of topic modelling in general produced by David Blei, Andrew Ng, and Michael Jordan in 2003 [9]. Given a collection of documents, it assigns to each topic a distribution over words and to each document a distribution over topics in an entirely unsupervised way.

LDA has been extensively used over these recent years, especially in the field of humanistic studies.

Relevant works include applications of LDA on news articles ([10], [11] and Rob Nelson's Mining the Dispatch [12], which looks at the changing discussion during the American Civil War); studies of the history of scientific ideas ([13], [14], David Mimno's computational historiography of classics journals [15] and Allen Riddell's study of German Studies publications [16]); topic-based search interfaces [17] and navigation tools for digital libraries [18].

Here we use LDA to identify topics on the text extracted from scraped web pages. This can give us two type of insights:

Topics allow us to understand on what areas of sustainability companies are focusing their action: environment, communities, supporting charities, employee well-being, etc.

The key assumptions of LDA are:

- Documents exhibit multiple topics (but typically not many). This is a distinguishing characteristic of LDA, in contrast with the assumption made by typical mixture models which assume documents exhibit only a single topic.
- With LDA all the documents in the collection share the same set of topics, but each document exhibits those topics in different proportion.
- In its topic distribution, each document, though it can use any of the topics, has only "activated" a handful of them.
- LDA is a probabilistic model with a corresponding generative process, the imaginary random process which explains how the observed words in documents are generated from an underlying latent structure.
- A topic is a distribution over a fixed vocabulary. These topics are assumed to be generated first, before the documents. The number of topics is specified in advance before any data has been generated.
- Order of words does not matter, bag-of-words approach

The next two sections describe the generative process assumed by LDA and the inference process to find topics in text documents.

5.1 The LDA Generative Model

LDA uses a generative model to explain how the observed words in a collection of documents are generated from an underlying latent structure.

In common words, the generative process is the following:

For each document in the collection, words are generated in a two-stage process.

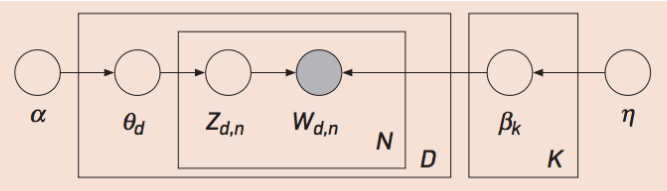
1. We choose a distribution over topics.
2. For each word in the document
    a. randomly choose a topic from the distribution over topics in step #1
    b. randomly choose a word from the corresponding topic (distribution over the vocabulary)

The same process is repeated for each document. Thus, every document is going to end up having different distributions over topics.

This statistical model reflects the intuition that documents exhibit multiple topics. Each document exhibits the topics in different proportion (step 1); each word in each document is drawn from one of the topics (step 2b), where the selected topic is chosen from the per-document distribution over topics (step 2a).

More formally, how the generative model works is described in the following directed graphical model, as formulated in [19]:

Figure 14: The LDA generative model. Source: Blei 2012



Each node here is a random variable.

The unshaded nodes - the topic proportions θ_d, assignments $Z_{d,n}$, and topics β_k - are the hidden variables.

α and η are the parameters of the respective Dirichlet distributions, which are represented by the two outer nodes in the generative model.

These two Dirichlet distributions are priors for two multinomial distributions which are parameterised by θ and β.

The Dirichlet distribution instead of being a distribution over events or counts (as most probability distributions are), it is a distribution over discrete probability distributions (eg. multinomials).

Another important note is that the Dirichlet distribution is the conjugate prior of the categorical distribution and multinomial distribution.

This means that when the Dirichlet distribution is used as a prior distribution for what we know about the parameters, and the information we get from the data follows a multinomial distribution, the posterior distribution again follows an (updated) Dirichlet distribution.

θ_d are the topic proportions for the d^{th} document, i.e. a distribution over topics for a given document d and since there are K topics, each $\theta_{d,k}$ represents the topic proportion for topic k in document d.

On the other side, β_k is the distribution over the fixed vocabulary for a given topic k and since there are K topics we can use the notation $\beta_{1:K}$ to indicate the topics.

And finally, Z_d is the topic assigned to for the d^{th} document, where $Z_{d,n}$ is the topic assignment for the n^{th} word in the document d.

W_d, the words of the d^{th} document, are shaded. This node is shaded to indicate that it is observed and not a latent variable, and given the vocabulary with fixed number N of words, $W_{d,n}$ is used to indicate the nth word in document d.

Given the above graphical model and notation, the generative process for LDA corresponds to the following joint distribution of the hidden and observed variables:

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}) = \prod_{i=1}^{K} p(\beta_i) \prod_{d=1}^{D} p(\theta_d) \prod_{n=1}^{N} p(z_{d,n}|\theta_d) p(w_{d,n}|\beta_{1:K}, z_{d,n})$$

Now, given that the general structure of how the documents are generated is known and it is also known how many times each word appears in each document (LDA uses a bag-of-words approach), it is possible to work backwards and find the most likely parameters that could have generated these documents.

This can be thought of as "reversing" the generative process: what is the hidden structure that likely generated the observed collection of documents? This is done through the process known as probabilistic inference.

5.2 Posterior computation for LDA

From the joint distribution shown above, the problem now turns into finding the conditional distribution (called the posterior) of the topic structure given the observed documents:

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D} | w_{1:D}) = \frac{p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D})}{p(w_{1:D})}$$

An illustration of the generative process and the problem of statistical inference is shown in the following figure, taken from [20]:

Figure 15: The generative process and the problem of statistical inference. Source: Steyvers, Griffiths, Probabilistic topic models.

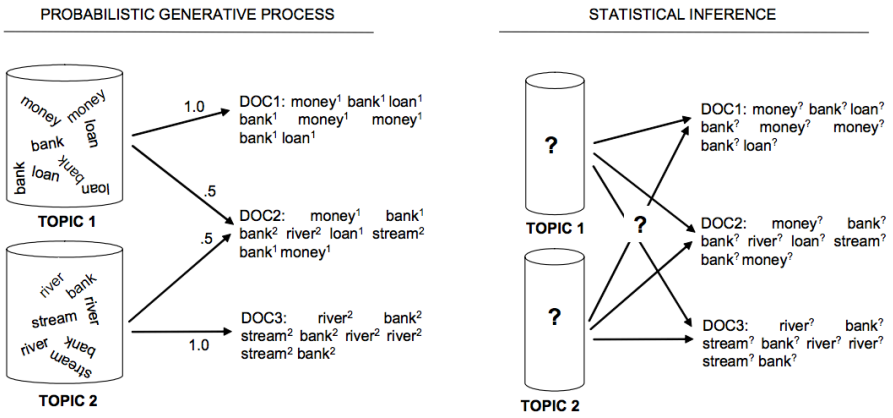


Figure 2. Illustration of the generative process and the problem of statistical inference underlying topic models

On the left, the generative process is illustrated with two topics. Topics 1 and 2 are thematically related to money and rivers and are illustrated as bags containing different distributions over words. Different documents can be produced by picking words from a topic depending on the weight given to the topic. Superscript numbers associated with the words in documents indicate which topic was used to sample the word. The right panel illustrates the problem of statistical inference. Given the observed words in a set of documents, we would like to know what topic model is most likely to have generated the data. This involves inferring the probability distribution over words associated with each topic, the distribution over topics for each document, and, often, the topic responsible for generating each word. [20]

One common technique to do so is to estimate the posterior of the word-topic assignments ($Z_{d,n}$), given the observed words, directly (whilst marginalising out β and θ). Inferring the word-topic assignments allows to easily derive the two unknown, hidden parameters, that represent the ultimate goal of LDA:

- β_k: the distribution over vocabulary for topic k
- $\theta_{d,k}$: the topic proportion for topic k in document d

The technique used is Gibbs sampling, originally proposed by Steyvers and Griffiths in [20], an example of a Markov Chain Monte Carlo (MCMC) technique. Markov chain in this instance means that we sample from each variable one at a time, keeping the current values of the other variables fixed.

The Gibbs sampler allows to infer the topic assignments Z from the observed data and produce the following estimate, where, using Steyvers and Griffiths notation:

- Z_i is the topic assigned to the i^{th} token in the whole collection;
- d_i is the document containing the i^{th} token;
- W_i is the word type of the i^{th} token;
- $Z_{\_i}$ is the set of topic assignments of all other tokens;

- · is any remaining information such as the α and η hyperparameters:

$$P(z_i = j|\mathbf{z}_{-i}, w_i, d_i, \cdot) \propto \frac{C^{WT}_{w_{ij}} + \eta}{\sum_{w=1}^{W} C^{WT}_{wj} + W\eta} \frac{C^{DT}_{d_{ij}} + \alpha}{\sum_{t=1}^{T} C^{DT}_{d_i t} + T\alpha}$$

where $\mathbf{C}^{WT}$ and $\mathbf{C}^{DT}$ are matrices of counts (word-topic and document-topic)

Once the sampling has converged, the document/topic distributions θ_d and topic/word distributions β_j can be trivially computed from the learned topic assignments, Z_i.

Finally, the results are

$$\beta_{ij} = \frac{C^{WT}_{ij} + \eta}{\sum_{k=1}^{W} C^{WT}_{kj} + W\eta}$$

(the probability of word type i for topic j) and

$$\theta_{dj} = \frac{C^{DT}_{dj} + \alpha}{\sum_{k=1}^{T} C^{DT}_{dk} + T\alpha}$$

(the proportion of topic j in document d).

5.3 Tuning the parameters of LDA

Selecting the number of topics is one of the most problematic modelling choices in finite topic modelling [21].

There is no clear method for choosing this parameter and the degree to which LDA is robust to a poor setting of it is not well-understood.

In practice, varying the number of topics tends to vary how "finely grained" the resulting topics are.

To decide the appropriate number of topics for the scraped pages a series of numbers were experimented: 5, 7, 10, 15 and 20.

For assessment and evaluation, the LDAvis [22] was used (see next Section). In the model with 5 topics, the resulted topics were broad and they overlapped in the 2-dimensional representation. As the number of topics were increased from 5 to 10 results were improving, and meaningful topics (words combinations) were emerging. In the solution with 20 topics, the impression was that the topics were picking out not consistent word combinations. The final choice was 15 topics.

In accordance with [Ref], the fact that the number of topics and the composition of the inferred topics can vary in this manner should reinforce the idea that an individual topic has no interpretation outside the particular model in use. This characteristic/limitation of LDA was already made clear by Blei and his coauthors in the orginal [9].

Another important parameter is α, the parameter of the document-topic distribution. As said already, documents exhibit multiple topics (but typically not many). This affirmation can be controlled by the α value.

A high α value means that each document is likely to contain a mixture of most of the topics. Conversely, a low α value means that it is more likely that a document may contain mixture of just a few of the topics.

The value of α was set to 0.001.

On the corpus and the related vocabulary two additional tunings improved the results:

- Removing trailing words: trailing words are words that appear few times in the whole corpus. Words that appear less than 2 times were removed.
- The removal of pages with < 5 words.

Because of this, 2 companies were dropped from the analysis.

5.4 The topics

The topics are here listed with the top 5 terms ranked in decreasing order according their topic-specific probability.

- **Topic 1:** 0.022 * environmental + 0.016 * sustainability + 0.013 * sustainable + 0.012 * management + 0.011 * project
- **Topic 2:** 0.017 * suppliers + 0.015 * safety + 0.011 * health + 0.010 * work + 0.009 * ethical
- **Topic 3:** 0.024 * energy + 0.016 * waste + 0.014 * carbon + 0.011 * year + 0.010 * emissions
- **Topic 4:** 0.027 * business + 0.017 * people + 0.012 * work + 0.012 * employees + 0.009 * community
- **Topic 5:** 0.013 * partners + 0.013 * local + 0.011 * noise + 0.009 * business + 0.009 * people
- **Topic 6:** 0.020 * business + 0.011 * waste + 0.010 * company + 0.009 * supply + 0.007 * health
- **Topic 7:** 0.022 * emissions + 0.018 * responsible + 0.013 * air + 0.012 * passengers + 0.011 * aircraft
- **Topic 8:** 0.034 * local + 0.020 * community + 0.013 * people + 0.013 * support + 0.012 * work
- **Topic 9:** 0.017 * charity + 0.013 * support + 0.010 * year + 0.010 * children + 0.009 * people
- **Topic 10:** 0.024 * waste + 0.011 * recycling + 0.009 * year + 0.007 * recycle + 0.007 * environmental
- **Topic 11:** 0.019 * social + 0.014 * wir + 0.013 * und + 0.010 * corporate + 0.009 * responsibility
- **Topic 12:** 0.019 * en + 0.019 * la + 0.017 * driving + 0.012 * wildlife + 0.010 * fuel
- **Topic 13:** 0.013 * environmental + 0.010 * car + 0.010 * waste + 0.009 * oil + 0.008 * seat

- **Topic 14:** 0.012 * sustainable + 0.011 * sra + 0.010 * ve + 0.010 * sustainability + 0.009 * caterers
- **Topic 15:** 0.009 * des + 0.008 * site + 0.008 * nos + 0.008 * construction + 0.008 * cent

5.5 Visualising topics as distributions over words

LDAvis [22] is a web-based interactive visualisation of topics estimated using LDA. It provides a global view of the topics (and how they differ from each other), while at the same time allowing for a deep inspection of the terms most highly associated with each individual topic.

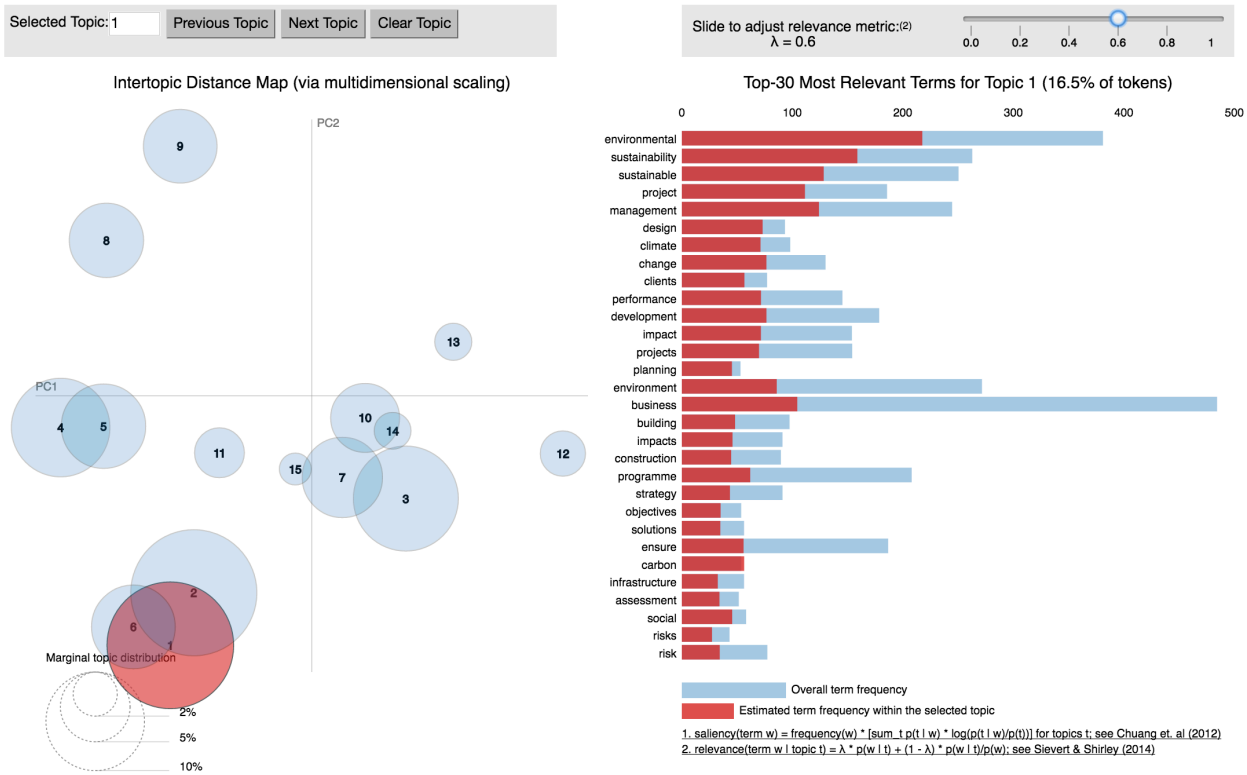
The visualisation (illustrated in Figure 16) has two basic pieces.

The left panel visualise the topics as circles in the two-dimensional plane whose centres are determined by computing the Jensen–Shannon divergence between topics, and then by using multidimensional scaling to project the inter-topic distances onto two dimensions, as is done in [23]. Each topic's overall prevalence is encoded using the areas of the circles.

The right panel depicts a horizontal bar chart whose bars represent the individual terms that are the most useful for interpreting the currently selected topic on the left. A pair of overlaid bars represent both the corpus-wide frequency of a given term as well as the topic-specific frequency of the term, as in [24].

The λ slider allow to rank the terms according to term relevance.

Figure 16: The resulting topic-terms distribution of the LDA model visualised using LDAvis.



5.5.1 Jensen–Shannon divergence

The similarity between topics is measured by calculating the similarity between their corresponding distributions over the fixed vocabulary. The similarity between topic distributions is computed using the Jensen-Shannon (JS) divergence, a popular method of measuring the similarity between two probability distributions.

Two distributions p and q are similar if they are similar to their average (p + q)/2.

5.5.2 Classical Multidimensional Scaling

Classical Multidimensional Scaling, also known as Principal Coordinate Analysis (PCoA), is a method similar to Principal Component Analysis (PCA). Rather than using raw data, PCoA takes a dissimilarity matrix as input. Like PCA, PCoA produces a set of uncorrelated (orthogonal) axes to summarise the variability in the data set. The result is the reduction of the original number of axes to a few of PCoA axes, in this case 2, which capture most of the variation of the dissimilarity matrix, and allow to project the intertopic distances onto two dimensions.

5.5.3 Terms ranking according to term relevance

By defaults the terms of a topic (on the right panel) are ranked in decreasing order according their topic-specific probability.

The ranking of the terms based on much discriminatory are for the specific topic, i.e. based on their "term relevance" to the topic, instead of the default setting, was originally proposed in [22].

Let kw denote the probability of term w ∈ {1, ..., V } for topic k ∈ {1, ..., K}, where V denotes the number of terms in the vocabulary, and let p_w denote the marginal probability of term w in the corpus. The relevance of term w to topic k given a weight parameter λ (where $0 \leq \lambda \leq 1$) is defined as:

$$r(w, k \mid \lambda) = \lambda \log(\phi_{kw}) + (1 - \lambda) \log\left(\frac{\phi_{kw}}{p_w}\right)$$

where λ determines the weight given to the probability of term w under topic k relative to its lift (measuring both on the log scale).

Setting $\lambda = 1$ results in the familiar ranking of terms in decreasing order of their topic-specific probability, and setting $\lambda = 0$ ranks terms solely by their lift.

In [22], the authors suggest that the "optimal" value of λ, obtained from a user study, should be 0.6.

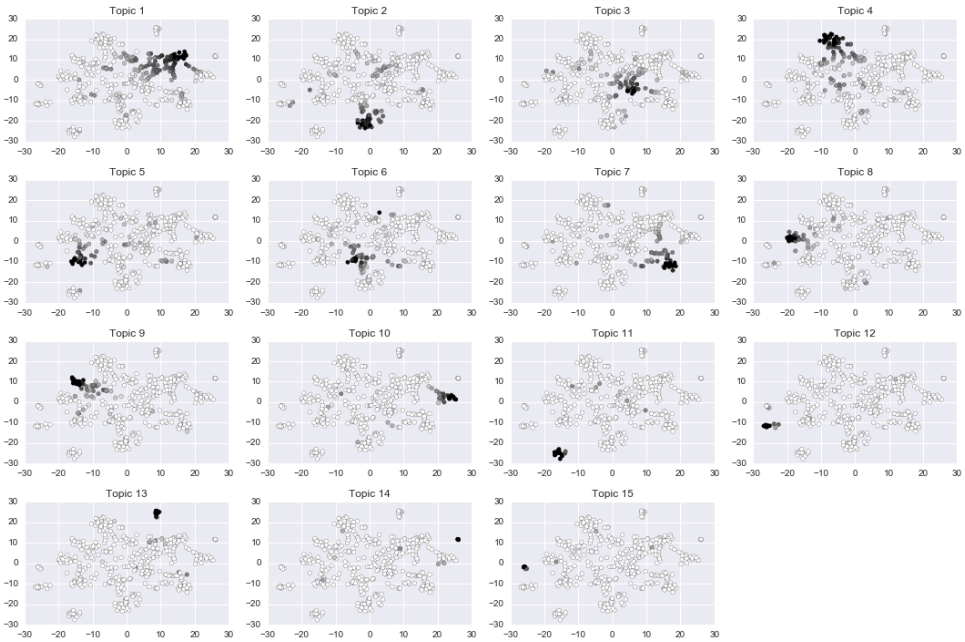
5.6 Visualising documents as distributions over topics

Other than producing the topic-word distribution, LDA also generates the document-topic distribution, i.e. the topic proportions for each document.

Each document is expressed as a 15-dimensional vector where each element represents the proportion of the document assigned to the specific topic. In Figure 17 documents are visualised as dots in a 2-dimensional plane, by using the t-

distributed Stochastic Neighbour Embedding (t-SNE) to project the inter-document distances onto two dimensions. The scatter plot is repeated for 15 times, as the number of topics. In each subplot, each document-dot is coloured according to the topic-specific proportion, on a scale from white, 0, to black, 1.

Figure 17: Visualising documents in a 2-dimensional plane. Each subplot shows the intensity of the specific topic, ranging from 0 to 1 (from white to black).



5.6.1 t-distributed Stochastic Neighbour Embedding

t-SNE is often very successful at exposing clusters in data by reducing the tendency to crowd points together in the centre.

t-SNE tries to optimise for preserving the topology of the data. It works by converting similarities of data points to probabilities. The affinities in the original space are represented by Gaussian joint probabilities and the affinities in the embedded space are represented by Student's t-distributions.

5.7 Industry patterns

To understand how the industry affects sustainability reporting, document-topic distributions are grouped by company and averaged.

These can be referred to as company-topic distributions.

This allows to compare the topic distribution of companies that belong to the same industry.

Companies that belonged to more than one industry were excluded. Also, companies for which the industry was not clear were not considered.

Companies that belong to the Oil & gas, Transport, Fashion Retailers and Construction industries were the most easily identifiable, thus the analysis that follows focuses on those industries.

Each plot is constructed as follows:

- Topic numbers are on the x-axis. This number is taken from the topics visualisation in Figure 16
- The y-axis tells the topic proportions
- Each line represents a company
- The most frequent terms for the peaks are included in the picture

Figure 18: Fashion Retailers

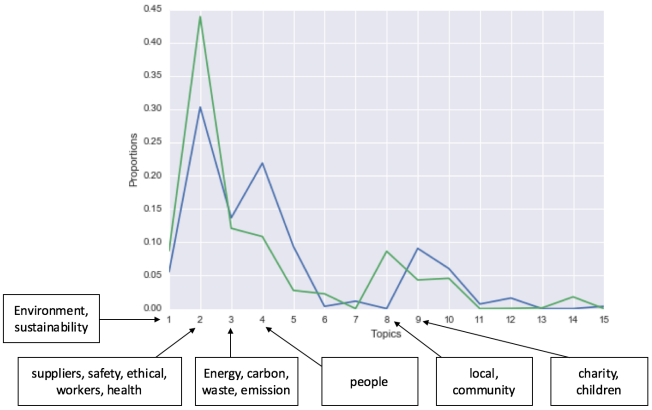


Figure 19: Construction

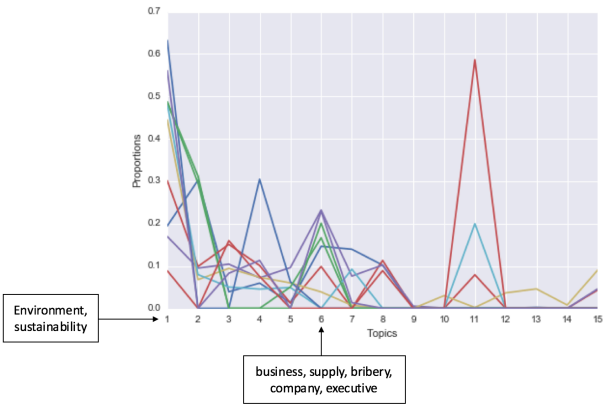


Figure 20: Oil & Gas

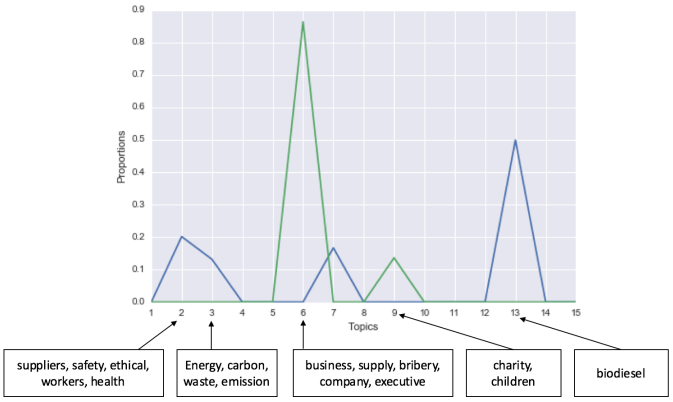
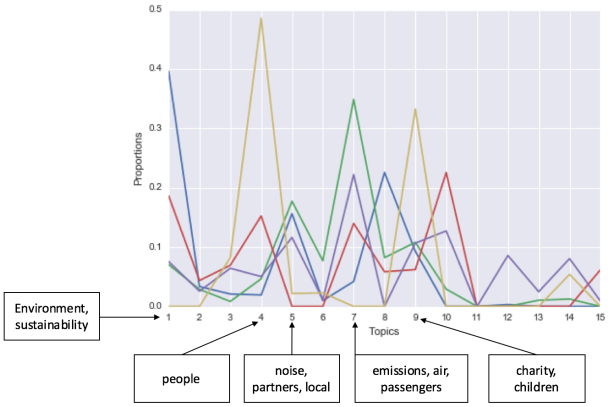


Figure 21: Transport



From the plots it is possible to see some patterns emerge. For example, in the construction industry, all companies talk about environment. Topic number 6, for which the most common terms are *business*, *supply* and *bribery*, is also popular.

The two fashion retailers split their distributions between topics that relate to the environment, and topics related to people, charities and the community.

Companies in Transport industry have more heterogeneous topics distributions.

Overall, this example shows how analysing companies web pages reveals that companies belonging to the same industry tend to focus their actions on related topics.

6 Conclusions

The results show that it is possible to discern the number of companies publishing sustainability information via scraping of their websites.

The scraper was developed bearing in mind that sustainability is a broad topic that can span from resource efficiency, environment protection and climate change, to assisting the local community, to improving worker life and many others.

In Section 2 the scraper was described in greater detail.

A special focus was given to the problem of extracting just the main text from the sustainability-related web pages. Four methods were benchmarked on a sample of 30 pages. The result of the test showed that smarter methods, which assign scores based on deterministic rules or use machine learning models optimised for the task, are better in recognising the "bad" content. However, they also tend to wrongly discard a varying proportion of "good" content. At last, the dragnet library, which uses machine learning to extract the right content, was chosen.

Overall, 563 sustainability-related web pages were scraped from 59 companies. The scraper could not access or did not find the sustainability content for 6 companies. For the remaining 35 companies, the scraper rightfully did not find any sustainability content as there was any published.

This result is a good starting point.

Future developments could be directed towards making the scraper more robust in terms of the keywords used to identify sustainability-related pages and the handling of specific obstacles, such as forms, pages translated into multiple languages, and so on.

However, the mere action of searching for keywords might not be sufficient in the context of Official Statistics, and here it is why this proof-of-concept went a step further, applying text analysis to the content of the pages to extract additional insights.

Section 4 illustrated the preprocessing of the text, a fundamental task to be performed before moving to the application of the Latent Dirichlet Allocation, the topic modelling technique used to identify topics in the web pages.

By tuning the model parameters, the topics became much clearer and consistent.

The topics offered a wider understanding of the different areas that sustainability covers and how this varied across industries.

The extent to which the numbers of sustainability reporting are affected by the industry and size of the company are worth exploring more, especially when a much larger number of companies will be considered.

The analysis of the text extracted from the web pages shows that the subject of sustainability is much more nuanced than what might result from a mere keyword analysis.

A Recommendations

The recommendations from the results drawn in this report are as follows:

Recommendation 1: Move to the IDBR as starting point.

The IDBR is a key data source for the analyses of business activities and production of related statistics, so it is logical that the formation of the SDG indicator should start from it. The larger sample of businesses extracted from the IDBR (~10,000) will give a much more complete view of the UK progress, and, because more documents are generated, the model will become more stable and patterns become clearer.

However, by not providing the website addresses, the IDBR introduces an additional step to be tackled before proceeding with web scraping.

Recommendation 2: Progress with the development of the scraper.

It is suggested that future developments could be directed towards:

- Revision of the keywords chosen to identify the sustainability content on web pages
- Distributing the scraper: let it run by several machines will help reduce the time required to collect data from a larger number of companies.
- Error handling: Scrapy handles some of the response errors already. However, an extensive error handling could be implemented separately to make the program more robust.
- Focused scraping: websites are organised in a tree structure and an idea could be that once a sustainability section is identified, the scraper focuses only in the subpages of that section, avoiding unnecessary scraping. It could be achieved by creating an ad hoc que management of the requests to be fired.
- Splash integration: many modern websites run entirely on JavaScript, and require scripts to run for the page to render properly. In many cases, pages also present dialogues that require a user action before showing the full page. For the time being Splash, a JavaScript rendering service to handle websites that use a heavy amount of JavaScript, has been successfully tested, but it has not been integrated into the final scraper.

Recommendation 3: Progress with the analysis of the textual content.

New research could be focused on:

- Real-time results: Spark could be used to digest the output of the scraper and achieve real-time preprocessing of the documents. Once preprocessed, the documents can feed directly into an an online implementation of the LDA model (i.e. the model constantly updates as new data comes in). As described in [Ref] Online LDA uses Online variational inference for learning an LDA model by processing the documents incrementally in small batches. It can handily analyse massive document collections, including those

arriving in a stream. The latter means that online LDA does not need to locally store or collect the documents: each can arrive in a stream and be discarded after one look. This obviously eliminates the waiting time for the scarper to finish before applying the LDA model. Potentially, it has also important implications on the ethical side of web scraping, because the action of "storing" the data can be avoided.

B Professional issues - Web scraping etiquette

Web scraping is defined as the mechanism through which a computer read a web page without the need of user intervention. Instead of presenting the data served by the website on a computer screen, the web scraping software directly grabs the content, or a subset of it, from the returned page.

Web scraping practices can fall into unethical territory in two main ways:

1. **The amount of burden imposed on the website.** This can take the form of the scraper reading the pages of a website much faster than a person could. If unexpected, the situation can cause difficulty for the servers to handle it and, ultimately, cause degraded performance for the rest of users of the website. Although, presumably, the web scraper is likely to be blocked before arriving at this. How much burden the web scraper can impose on a website vastly depends on several factors, such as how much traffic the website normally handles.
2. **How the data extracted are used.** This aspect is often the more debated, as there is no clear distinction between legal and illegal web scraping.

Among web scraping developers, exists a series of widely recognised thumb rules, an 'etiquette', to follow for a polite behaviour.

The following steps were taken to comply with the web scraping etiquette:

- **Limiting the crawl depth to 3:** the maximum depth that will be allowed to crawl for any site.
  Websites contain multiple pages, which in turn can contain subpages.
  A website's homepage has a crawl depth of 0. Pages linked directly (with one click) by the homepage have a crawl depth of 1; pages linked directly by depth-1 pages have a crawl depth of 2, and so on.
  The default setup of the parameter controlling the depth limit in Scrapy has value 0, which corresponds to no limit imposed.
  In this project the depth limit is 3, considered an exhaustive depth for the purpose of this project.

- **Download delay set to 5:** the number of seconds that the downloader should wait between every consecutive page downloaded from the same website.
  The download delay parameter can be used to avoid hitting servers too hard.
  By default, the download delay is 0. For this project, the download delay is 5. Scrapy will take care to use a random interval from 2.5 (0.5*5) and 7.5 (1.5*5) seconds between every request.

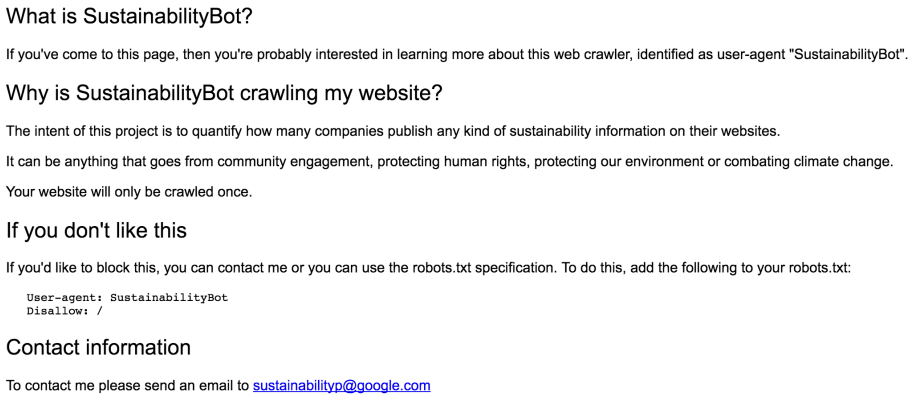
- **Setting a meaningful User Agent:** the User Agent string is the way the web browser tells a website information about the browser and operating system accessing the web page.

This information allows the website to customise content for the capabilities of a particular device.
Given the fact that the scraper is not a browser but a script, the user agent field can be used instead to include a URL where the website administrator may find out more information about this project. The user agent string set in this project:

SustainabilityBot `http://goo.gl/4VlxkT`

which points to this page to know more about the project:

Figure 22: Page created to give information about the project



What is SustainabilityBot?

If you've come to this page, then you're probably interested in learning more about this web crawler, identified as user-agent "SustainabilityBot".

Why is SustainabilityBot crawling my website?

The intent of this project is to quantify how many companies publish any kind of sustainability information on their websites.

It can be anything that goes from community engagement, protecting human rights, protecting our environment or combating climate change.

Your website will only be crawled once.

If you don't like this

If you'd like to block this, you can contact me or you can use the robots.txt specification. To do this, add the following to your robots.txt:

```
User-agent: SustainabilityBot
Disallow: /
```

Contact information

To contact me please send an email to sustainabilityp@google.com

- **Respecting robot.txt:** website owners use the robots.txt file to give instructions about their site to web robots (such as web scrapers).
  A robots.txt is a file that can be accessed, when present, at the root of a website by simply specifying the root URL followed by /robots.txt:

  http://www.example.com/robots.txt

  robots.txt files use the Robots Exclusion Standard[32], a protocol with a small set of commands that can be used to indicate what can be accessed on the website and by who.
  The instructions in robots.txt files cannot force a robot to follow the proposed behaviour; instead, these instructions act as directives.

  As an example, this is the INEOS robots.txt file which can be accessed at `http://www.ineos.com/robots.txt`

[32] `http://www.robotstxt.org/`

Figure 23: INEOS robots.txt file

```
# www.robotstxt.org/

# Allow crawling of all content
User-agent: *
Disallow: /Static/IneosStatic/dist/
```

Lines starting with “#” are comments.
"User-agent: *" indicates that what follows applies to all robots.
The keyword "Disallow" tells the robots listed in the previous line (in this case is for anyone) that they should not visit any of the pages on the INEOS website listed after it.

This web scraping project respects robots.txt policies.
By enabling this option in the scraper configuration file, Scrapy takes care automatically of the interpretation of the robots.txt files.

C Self-assessment

This project allowed me to learn in a much deeper detail what is web scraping and how Scrapy works, and to scratch the surface of the vast area of Natural Language Processing.

I have been leading this project since the very beginning, from the definition of the objectives, the formal draft of the plan, to the concrete implementation and the final recommendation here included.

The idea for the project was developed in response to the ONS SDG team request to assess the feasibility of using the Web as the source for creating a new indicator. The Request came in March. The next month, I worked in collaboration with the ONS Admin Data Division and my senior manager to write a recommendation paper for the SDG team. The paper considered Web-data together with other sources, evaluating pros & cons. The recommendation paper was integrated with a preliminary plan for the project. Technical challenges along with ethical considerations and costs were carefully considered.

The project formally started in July as an internal proof-of-concept, together with my upgrade to the role of Research Officer within the Big Data team.

In the two months and a half that have followed, my supervisor and several colleagues helped me questioning ideas, giving me precious hints to overcome the problems and challenges this project demonstrated since the very beginning.

Overall, it has been a challenging and rewarding experience, with a good mix of project management, collaboration and technical skills. It allowed me to cover several data science aspects. I had to make choices for the tools to use (programming language, where and how to store the data, the Scrapy framework). The preprocessing of the text and the tuning of the topic modelling algorithm took a substantial amount of time.

Lastly, deriving the insights in the final part. I included in this report my final recommendations, with some exciting ideas for future developments.

D How to Use My Project

In the dissertation directory there are 7 main subdirectories:

1. **scraper:** contains the scraper. The main files are:
   - **seeds.txt :** the starting URLs
   - **items.py:** contains the Item class which specify the Item fields
   - **pipelines.py**: is the file that contains the two components of the Item pipeline, the ContentExtractor and the MongoConnector
   - **settings.py:** specify the settings for the scraper
   - **web.py:** the scraper

   The scraper need to be run from the terminal positioned in the scraper directory. Also, MongoDB needs to be up and running so that the scraper can establish the connection.

   The command to run the scraper is:

   ```
   scrapy crawl SustainabilityBot
   ```

2. **notebooks:** to write the code the IPython[33] notebook has been used. There is a file for each main section written in this report:
   - **top_companies.ipynb:** covers the section of acquiring the strting URLs from the fasttrack website.
   - **extraction_methods.ipynb:** covers the comparison of the four methods for extracting the text out of the webpages
   - **websites_stats.ipynb:** contains the statistics calculated from the webpages extracted by the scraper
   - **text_preprocessing.ipynb:** preprocess the text extracted from the webpages. Also create the corpus and the dictionary
   - **text_analysis.ipynb:** load back the corpus and the dictionary just created and run the LDA algorithm. It visualises the topic-term distributions, the document-topic distributions and the industry patterns

   Additional files included are:
   - **stoplist.txt:** the list of stopwords used
   - **util.py:** contains some helper functions

3. **notebooksHTML:** the IPython notebook was chosen because it offers the possibility to download the compiled code with the output results in the form of a HTML page. Textual paragraphs are included to aid the understanding of the code. This folder contains one HTML file for each IPython notebook in the *notebooks* folder

4. **corpus:** contains the corpus and the dictionary created from the scraped web pages

[33] `https://ipython.org/notebook.html`

5. **lda:** contains the final LDA model

6. **images:** contains all the images generated from the notebooks

7. **docs:** contains the page created to find more information about the SustainabilityBot. Live here `http://goo.gl/4VlxkT`

References

[1] United Nations Statistics Division, "Final list of proposed Sustainable Development Goal indicators," June 2016. [Online]. Available: http://unstats.un.org/sdgs/indicators/Official%20List%20of%20Proposed%20SDG%20Indicators.pdf.

[2] United Nations, "Goal 12: Ensure sustainable consumption and production patterns," [Online]. Available: http://www.un.org/sustainabledevelopment/sustainable-consumption-production/.

[3] United Nations General Assembly, "2005 World Summit Outcome," 24 October 2005. [Online]. Available: http://data.unaids.org/Topics/UniversalAccess/worldsummitoutcome_resolution_24oct2005_en.pdf.

[4] D. Kouzis-Loukas, Learning Scrapy, Packt Publishing, 2016.

[5] M. E. Peters and D. Lecocq, "Content Extraction Using Diverse Feature Sets".

[6] C. Moral, A. de Antonio, R. Imbert and J. Ramírez, "A survey of stemming algorithms in information retrieval".

[7] M. Porter, "An algorithm for suffix stripping".

[8] C. D. Paice, "Another Stemmer".

[9] D. M. Blei, A. Y. Ng and M. I. Jordan, "Latent Dirichlet Allocation".

[10] D. Newman, C. Chemudugunta, P. Smyth and M. Steyvers, "Analyzing entities and topics in news articles using statistical topic models".

[11] D. J. Newman and S. Block, "Probabilistic Topic Decomposition of an Eighteenth-Century American Newspaper".

[12] R. K. Nelson, "Mining the Dispatch".

[13] D. Blei and J. Lafferty, "A correlated topic model of Science".

[14] D. Hall, D. Jurafsky and C. D. Manning, "Studying the history of ideas using topic models".

[15] D. Mimno, "Computational Historiography: Data Mining in a Century of Classics Journals".

[16] A. Riddell, "How to Read 22,198 Journal Articles: Studying the History of German Studies with Topic Models".

[17] D. o. C. S. U. o. M. IESL, *http://rexa.info/.*

[18] K. Lerman, *Document clustering in reduced dimension vector space.*

[19] D. Blei, "Probabilistic Topic Models".

[20] M. Steyvers and T. Griffiths, "Probabilistic Topic Models".

[21] H. M. Wallach, D. Mimno and A. McCallum, "Rethinking LDA: Why Priors Matter".

[22] C. Sievert and K. E. Shirley, "LDAvis: A method for visualizing and interpreting topics".

[23] J. Chuang, D. Ramage, C. D. Manning and J. Heer, "Interpretation and Trust: Designing Model-Driven Visualizations for Text Analysis".

[24] J. Chuang, C. D. Manning and J. Heer, "Termite: Visualization Techniques for Assessing Textual Topic Models".