DATA COLLECTION ON THE INTERNET FOR SEWAGE SYSTEM CARTOGRAPHY

Project n°6 – Summary report

ARTIGUES Gabriel & LABAT Coline
Projet Industriel de Fin d'Etudes IG5
Polytech Montpellier
From 28th November 2016 to 9th February 2017

For Cart'Eaux project and Berger-Levrault enterprise,
For the attention of Mrs DELENNE Carole, Mrs CHAHINIAN Nanée and Mr DERUELLE Laurent
Supervisor: Mrs LAURENT Anne
Examinations board composed of Mrs TOULEMONDE Gwladys and Mr CHAPELLIER Philippe

Berger Levrault
L'AVENIR EST AUX VALEURS SÛRES

IG

POLYTECH MONTPELLIER

UNIVERSITÉ DE MONTPELLIER

Table of contents

Table of figures

ACKNOWLEDGMENTS

First, we would like to express our sincere gratitude to all the Cart'Eaux project members for offering us this work on their project.

We would like to thank Carole DELENNE and Nanée CHAHINIAN for their trust, their time, their advice and their availability all along this project.

We would also thank Benjamin COMMANDRE for his availability and his help on the word analysis.

Finally, we would like to thank Anne LAURENT, our tutor, for her attentiveness, her advice and her support all along our mission.

INTRODUCTION

Nowadays, municipalities in France have expertise in terms of sanitation and they have to establish a descriptive detail of collection work and wastewater transportation. Although maps and geographic data are more frequently digitized, they are, for a big part, still in analog format, that makes it difficult to use and update. Attributes and characteristics linked to different objects that form the network are, a priori, available. We find it in public or trade databases (which are often incomplete and updating is uncertain), in request for proposals, intervention report, press articles doing damages statement… Information dividing up makes networks accurate cartography and hydraulic design complex and tiresome.

The Cart'Eaux project aims at developing a multi-sources data collection method to merge knowledge into information in order to allow the cartography and modelling of urban wastewater and stormwater networks.

The Cart'Eaux project is divided into 4 distinct parts:

1. Creation and compilation of geographical data
2. Creation of the attribute table
3. Wastewater network cartography
4. Hydraulic simulation

We are involved in action 2 to automatically collect documents on the web.

> *« Action 2. Création de la table attributaire*
>
> *Internet recèle de nombreux documents, rapports publics et web services susceptibles de contenir une description des interventions sur les réseaux (e.g. travaux, entretien, réparation). Cette deuxième action propose de mettre en œuvre des techniques de fouille de données sur internet pour découvrir ces documents de différents formats (texte html, pdf, images, plan numérisé) afin d'identifier et de consolider un maximum d'attributs liées aux objets, incluant un indice de confiance. L'open data et les services mis en place par la communauté d'agglomération de Montpellier dans le cadre du projet « smart cities » pourront servir à alimenter cette table attributaire. […] »* ***[1]***

Our work is important for the Cart'Eaux project because it will enable an automated processing chain in order to perform data mining on the documents found with our solution.

This report highlights our approach in the comprehension of the mission proposed by the Cart'Eaux project members. It was important for us to define it in the most detailed way in order to precisely meet their expectations. Technical information is included in our technical report.

I. MISSION CONTEXT

a. Cart'Eaux project and Berger-Levrault company

The Cart'Eaux project aims at making wastewater network cartography easier in order to have a better management of the urban underground space. It started on the 20th November 2015 and it is funded by the European Union via ERDF[1] funds.

Berger-Levrault is a company specialized in software publishing. It has local authorities for main target and offers them the necessary services for their water and sanitation expertise.

b. Team

All through the project, we met, once a week, the Cart'Eaux project members which were affected to different tasks in this project:

- Carole DELENNE, who supports the Cart'Eaux project, is senior lecturer at Polytech Montpellier and HydroSciences Montpellier. She is also in charge of action 4 of the project: "Hydraulic modelling".
- Nanée CHAHINIAN who is a hydrologist and research fellow at IRD[2] assigned to HydroSciences Montpellier. She is in charge of action 3 of the project: "wastewater network cartography".
- Benjamin COMMANDRE, is a computer science engineer at HydroSciences Montpellier. He is in charge of the development of an algorithm for network reconstruction, and the integration of the different modules developed in an automated electronic processing chain.
- Sandra BRINGAY, Full Professor at Paul Valéry University in Montpellier, working for LIRMM[3], she works as an expert in text analysis on the Cart'Eaux project.

We also met Laurent DERUELLE in charge of scientific projects at Berger-Levrault, and responsible for action 2 of the Cart'Eaux project: "attribute table creation".

c. Context

During an internship, a student implemented a program that conducts a search of information on French text documents:

> « *[...] The second phase of the process is then implemented as follows [...]:*
>
> - ***Thematic identification:*** *we established a list of words related to the field of interest, composed of [...] words which describe objects such as manhole covers and stormwater or wastewater networks, and [...] terms that depict their characteristics (e.g. diameter, depth, flow) and measurement units (e.g. m 3/s , Eq-hab, m). We completed this list with terms extracted from two lexicons (UNESCO, 1992;*

[1] ERDF : European Regional Development Funds
[2] IRD : Institut de Recherche pour le Développement
[3] LIRMM : Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier

Andréassian et al., 2002) and one reference work (Bourrier, 2008). […]

- ***Spatial identification:*** *[…] in order to recognize spatial information (e.g. "au nord de la route R12 allant de Montpellier à Lunel", "le boulevard du Languedoc").*

- ***Temporal identification:*** *[…] to detect time expressions (e.g. "l'appel d'offre signé du 12 mai", "du 03/06/14 au 21/07/14") »* ***[2]***

However, all the documents are collected by expert hydrologists manually, one by one.

d. Issues and goals

The collection of documents takes a lot of time to people affected to this task. Indeed, expert hydrologists have to search and find documents that could be interesting and contain information related to stormwater and wastewater networks. Then, they have to turn documents, that could be PDF documents, html pages, pictures or digitized maps, into text documents.

If this task was automated, it could save a lot of time and energy to all the Cart'Eaux project members. This would provide them a big enough collection of documents related to the subject and they would only have to focus on the textual analysis results. Then, thanks to word analysis results and the different documents as digitize map or picture, the collected data would be summarized in a hydraulic network model in order to enable the numeric simulation of flows in sewage systems.

e. Our mission

Our main mission was to develop a solution to find documents on the web related to wastewater and stormwater networks.

The first step of the mission was to give information about what was possible to do, technically and legally. Then, we had to find a way to process automated searches and to form our queries. Mrs. DELENNE and CHAHINIAN gave us a list of French keywords linked to the subject to be included in our queries.

Another optional mission was to present an automatic tool of news watch on some cities websites in order to see updates about wastewater and stormwater networks.

II. WHAT SOLUTIONS ARE POSSIBLE?

The demands were pretty clear but also complex because we had never worked on a data collection program before. We first conducted research on the possibilities offered to us in legal and technical terms. Some solutions recommend the using of a robot, so we went into this eventuality in depth.

a. Web crawler definition

> *« A Web crawler is an Internet bot which systematically browses the World Wide Web, typically for the purpose of Web indexing (web spidering). »* ***[3]***

It is conceived to collect resources (html pages, images, videos, Word, PDF or PostScript documents, etc.) to allow search engine to index them.

b. Regulations

Concerning the regulations, there are some legal limits. Indeed, robots should not be programmed to spam, to hack rivals websites, to pick up information about people, to appropriate information thanks to rivals.

The best thing to do is to follow operating search engine robots. Indexable websites are written in a robot.txt file which is first opened to check the web pages not to focus on during the exploration. Then, it is important to guarantee in the general terms and conditions that it is possible to explore the website.

There are also technical limits for the utilisation of a robot. The target website may block the IP address on which the robot is running in case of excessive utilization, we should store data used in a database. Furthermore some websites use captcha (picture that cannot be read by robots), and it is difficult to run robots on a website which uses cookies. ***[4]***

c. Tools to develop a robot

The open-source framework Scrapy is dedicated to crawl websites and structured extraction in webpages. It operates on most of the operating systems because it is developed in Python. Otherwise, it can assemble extracted data in JSON documents or in a MongoDB database for example.

The problem is that we have to run the search on URLs provided a priori. But, in our case, we had no predefined sources, the objective is to find sources linked to a specific studied topic, in order to extract relevant documents. A solution may be to give Scrapy a link as an input that corresponds to search results of a Google query, formed by the words “Prades-le-Lez canalisation assainissement”.

However, to crawl on a Google domain may turn out very complicated. Actually, in an ideal world, we should run several queries on Google formed by word combinations belonging to the provided vocabulary. This causes an answering time problem that we could resolve by running several servers in parallel. Moreover, Google is capable of blocking IP address if there are an unusually high number of queries in a short period.

Finally, one solution is to send queries on Google through a browser and then parse the html source code from the results page. But, reading the file https://www.google.com/robots.txt , it seems that crawling the path /search is forbidden and it will transgress the Google terms of service.

> *« You agree not to access (or attempt to access) any of the Services by any means other than through the interface that is provided by Google, unless you have been specifically allowed to do so in a separate agreement with Google. You specifically agree not to access (or attempt to access) any of the Services through any automated means (including use of scripts or web crawlers) and shall ensure that you comply with the instructions set out in any robots.txt file present on the Services. »* ***[5]***

d. Google Custom Search

In order to avoid problems and be legal, an alternative to robots is possible. Google provides a tool named Custom Search that enables the users to create a search engine for their website or their blog. This search engine can be customized in order to have an effect on the ranking, the look and feel of the search results. It can also be configured to search only the contents of one website, or to focus on a particular topic from multiple websites. But what makes it even more interesting is that Google makes a JSON API available that allow us to display and retrieve search results programmatically. With this API, we can use RESTful requests to get search results in JSON format. They are reachable through this URL:

https://www.googleapis.com/customsearch/v1?key=***KEY***&cx=***CX***&q=***QUERY***

The “key” parameter is the API key that you can obtain from the Google API Console and the “cx” parameter is the search engine ID created thanks to the Custom Search Engine. Finally, “query” is where you define the keywords for the web search. You can refer to the technical report to have more details on obtaining these elements. Google offers a free version that allows running 100 search queries by day. However it is possible to run more queries on a day, and it will cost 5$ per 1000 queries, up to 10k queries per day. ***[6]***

In order to discover this solution, we started some queries on our Google Custom Search Engine.

During our searches, we noticed that the results are different if the requests are done using Google Web Search on www.google.com or the Custom Search Engine (CSE). Indeed, CSE can search across a set of sites we specified or across the whole web. In the second case, CSE does not include Google Web Search features such as Oneboxes, real-time results, universal search… Also, the JSON API allows us to retrieve the 10 first results, which are considered as the 10 best results.

e. Conclusion

Other search engines provide such APIs like Bing Search API or Yahoo BOSS Search API with a bigger number of free queries, but they all have been discontinued. On the other hand,

Faroo offers a technology providing a Free Search API with 1 million free queries per month. The big drawback is that it gives results only in English, German and Chinese.

Consequently, we consider that using the Google CSE is the best solution, this is why we have chosen to integrate this solution in a program.

III. TECHNOLOGICAL CHOICE

To be the most efficient during the project, and in order not to waste time in the middle of the project in case the chosen technology is not suitable with the next step, we carried out some research on different technologies with their advantages and their drawbacks.

Tool	Advantages	Drawbacks
Java	Easy reusability of the code Several tasks performed at the same time	Speed of the program not really fast
Java Enterprise Edition	Execution in every OS[4] Already used for the web application	Complex architecture High competence level
Python	Existing tools to transform documents into text documents	Slow

Python allows us to use a lot of libraries like a tool to convert PDF documents in text document, a tool to retrieve html source code and convert it in text document, etc. By searching examples of web scraping, we found out that Python is the most used technology and that it is well suited to this kind of project.

Thanks to this search, we have chosen to use the Python language.

[4] OS : Operating System

IV. SOLUTION DEVELOPMENT AND TESTS

After our searches, we have chosen to use CSE for our solution and to work in Python. We defined steps to work on one after another. When the step was completed, we started working on the next step.

The aim of the program is to automatically throw a large number of queries through our Custom Search Engine supplied by Google, get the results under JSON format thanks to the API and then save locally the content of each result in text documents that can be processed later by the word analysis program.

Here are the main steps that we followed.

a. Model for queries

The first important task was to define how to model our queries to find the best way to collect documents that are the most related to the subject.

First of all, we did some search on the operators. Indeed, there are some operators that allow users to make searches and results deeper. It is possible to restrain search on webpages or domains by the operator **site:**, to exclude search results that include a specific word thanks to –, to find pages that contain only one term among all with **OR**, etc. Unfortunately, there is no operator that permits to force search results by the inclusion of a specific word. In our case, it would be the name of the city.

The Cart'Eaux project members gave us a list of keywords linked with stormwater and wastewater. After some tests, we concluded that the best method is to associate the name of the city, two keywords and a type of document in the query to find the maximum of results linked to the subject, without using search operators.

Indeed, if we give only the name of the city and a keyword like "collecteur", we can find information on garbage collection or collection in post offices. If the keyword is "conduite", the results deal with driving school. But if we make another search with the city and both keywords, it reduces the topics of the results and it makes the results more likely to deal with stormwater systems and wastewater.

Thanks to a sample of documents that were collected manually by the project members for the word analysis step, we defined some categories of document types. The different types allow us to find directly results that redirect towards some specific documents about the topic.

We chose to put keywords related to wastewater and stormwater systems and keywords concerning types of documents in two separated text documents. In agreement with the project members, we decided to create all the possible combinations of two stormwater or wastewater related keywords. Some of these associations were deleted because of the close meaning of the keywords.

Here are the two text documents that we have.

Wastewater and stormwater network keywords	Type of documents keywords
Collecteur Conduite Collecteur Canalisation Collecteur Pluvial Collecteur Assainissement Collecteur Réseau Séparatif … Canalisation Pompe … Pluvial Fossé …	annonce ville cahier des clauses techniques plu rapport annuel dossier enquete publique …

We have 393 combinations of technical keywords and 18 keywords for the type of documents.

b. Running queries

In our program, we first ask the user on which city he wants to perform his search. Consequently, the name of the city will be included in each query that we run. Then, the program associates each combination of keywords with a document type by reading each line from both text documents mentioned above. This makes a total of $393 \times 18 = 7074$ queries.

For executing the query, the program opens the URL:

https://www.googleapis.com/customsearch/v1?key=***KEY***&cx=***CX***&q=***QUERY***

At each iteration, the query parameter is replaced by the name of the city and the keywords read in the text documents. From this URL, we get to the results in a JSON format thanks to the Google API.

c. Results in JSON

For each query, we end up with a JSON document which is saved locally. In these documents, we can find general information about the search terms of the query, the total number of results, the number of results displayed (maximum 10) and for each result we can find the URL, the title of the webpage, a snippet, etc.

If the link refers to a PDF document, we can also obtain its creation and modification dates.

d. Collect documents from JSON

Once the JSON document is saved, we read and save some objects and their attributes in variables in order to reuse them in the program. Some will be used for the generation of the report (see below). The URL of each result is parsed to determine the extension of the result (html, PDF, php, etc.), it is programmatically visited and the source code of the webpage is locally saved in accordance with the extension previously defined: PDF documents will be

stored in PDF format, and for the others the source code will be stored. They are named with the couple `[display link]titleOfTheWebpage` in order to have a unique identifier and avoid duplication. For some results, we cannot determine the format because the extension is not specified in the URL. This concerns webpages from Wikipedia (https://fr.wikipedia.org/wiki/Eau_pluviale) or homepages (https://www.fransbonhomme.fr/) that do not contain relevant information on our topic. So we chose not to keep these sources.

e. Transform documents into text files

Now that we have the "source" documents, we can initiate the last step: convert them into text files to submit them to word analysis.

According to the format, a different library is used:

- PDFMiner for converting PDF files into text documents;
- html2text for transforming html, php and asp code into text documents.

They are named the same way as their source document. Only the extension changes: it is now `.txt`.

f. Report generation

After some meetings, the Cart'Eaux project members wanted a report of all the results from the queries. This summary, in text format, is generated each time the program is started. It is composed of essential information. For each new query, the user can find:

- The search terms (city + combination of sewage system keywords + type of document keyword)
- The total number of results
- The number of results displayed
- The list of the results with the URL, the name, the snippet and the format for each result.

This information is retrieve thanks to the JSON document. You can see an example of the report document below. Results with no identified format are still in the report if the members need to find them.

```
################################################
#############   Nouvelle requete   #############
################################################

Mots clés de la requete : prades le lez Collecteur Canalisation convention
d'application
Nombre total de résultats : 15
Nombre de resultats affichés : 10

Liste des résultats

URL : http://www.montpellier3m.fr/sites/default/files/reglement_veolia.pdf
Nom : Règlement_du_Service_Assainissement_|_VEOLIA
Résumé : contrat d'abonnement au Service de l'assainissement. Ce peut ... Montpellier ;
Prades le lez ; Pérols ; Saint Jean de Védas ;. Vendargues. .... le diamètre de
toutes les canalisations en domaine privé, ... piquage sur le collecteur principal
doit alors être réalisé ..... la Collectivité. Les modalités d'application de cette
redevance.
Format : .pdf
Date et heure de création : 24-02-2015 15:15:50
Date et heure de modification : 24-02-2015 15:15:50
```

Figure 1 – Part of a report document

g. Classification of files

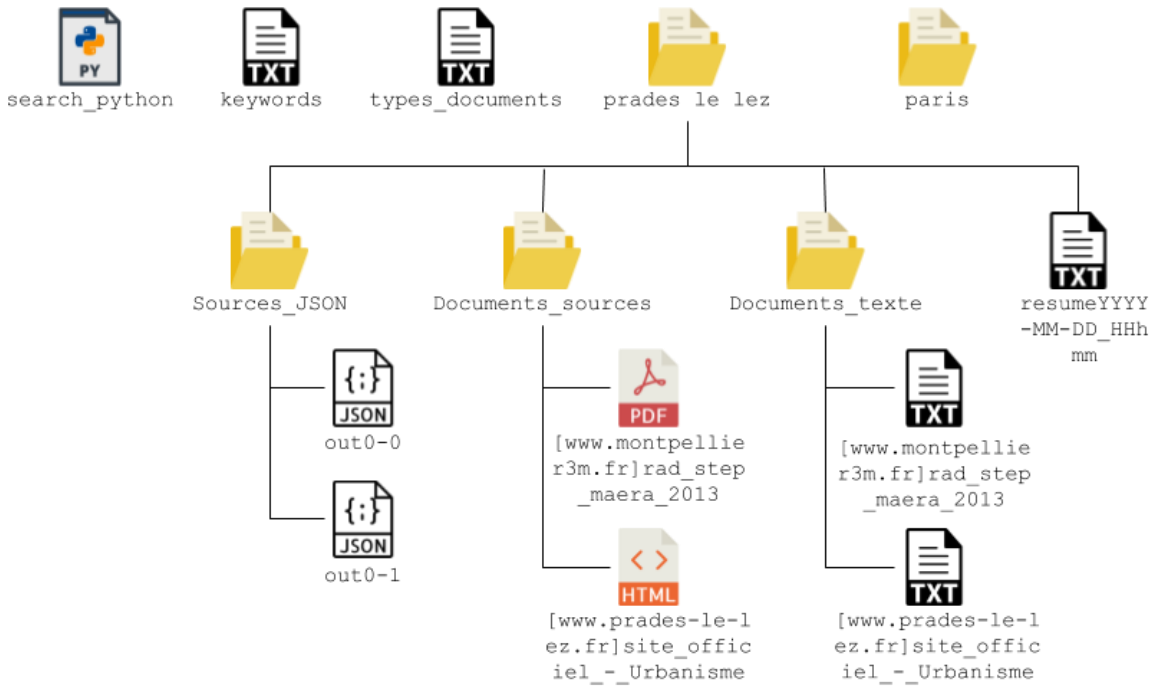


Figure 2 – Tree view of files

At the root, there is our python program and the text files for the keywords. When the program starts, a folder is created with the name of the city that the user specified. Then, in this folder, the user can find the report about the results (named with the execution date and hour) and three different folders:

- `Sources_JSON` where we can find the JSON documents,

- `Documents_sources` where the documents under their source format are stored,
- `Documents_texte` where the documents are saved in text format.

h. Pause and restart of the program

As the API allows 100 queries per day, the program needs over 70 days to be executed completely (7074 queries). This threshold must not be exceeded, otherwise an error comes out and stops the program. In order to avoid this issue, each time a query is run, a counter is incremented. When 100 queries are done, the program is paused during 24 hours, so as the Google account recovers its quota. During this period, the user can access the documents already generated. 24 hours later, the program starts again at the same point it was stopped.

At the end of the execution, once all the queries have been run, an email is send to the user in order to warn him. Attached to this email, the user can find the report document concerning all the results.

i. Tests

Each functionality that we have implemented has been tested. In other words, each time we made a modification, we ran the program on a small sample of queries. Only when it worked we went on to the next step. Consequently, we checked every time that all our already developed functionalities were not affected by the new one.

Also, we ran the program on a large period of time to test that it was capable of stopping and restarting itself and that there were no errors to interrupt it. We executed it on several cities to make sure that documents collected will be different and in order to handle the maximum of errors.

In order for the user to follow the progress of the execution, we display information on the console all along the running. Therefore, he can see the steps and be aware of the good continuity of the program.

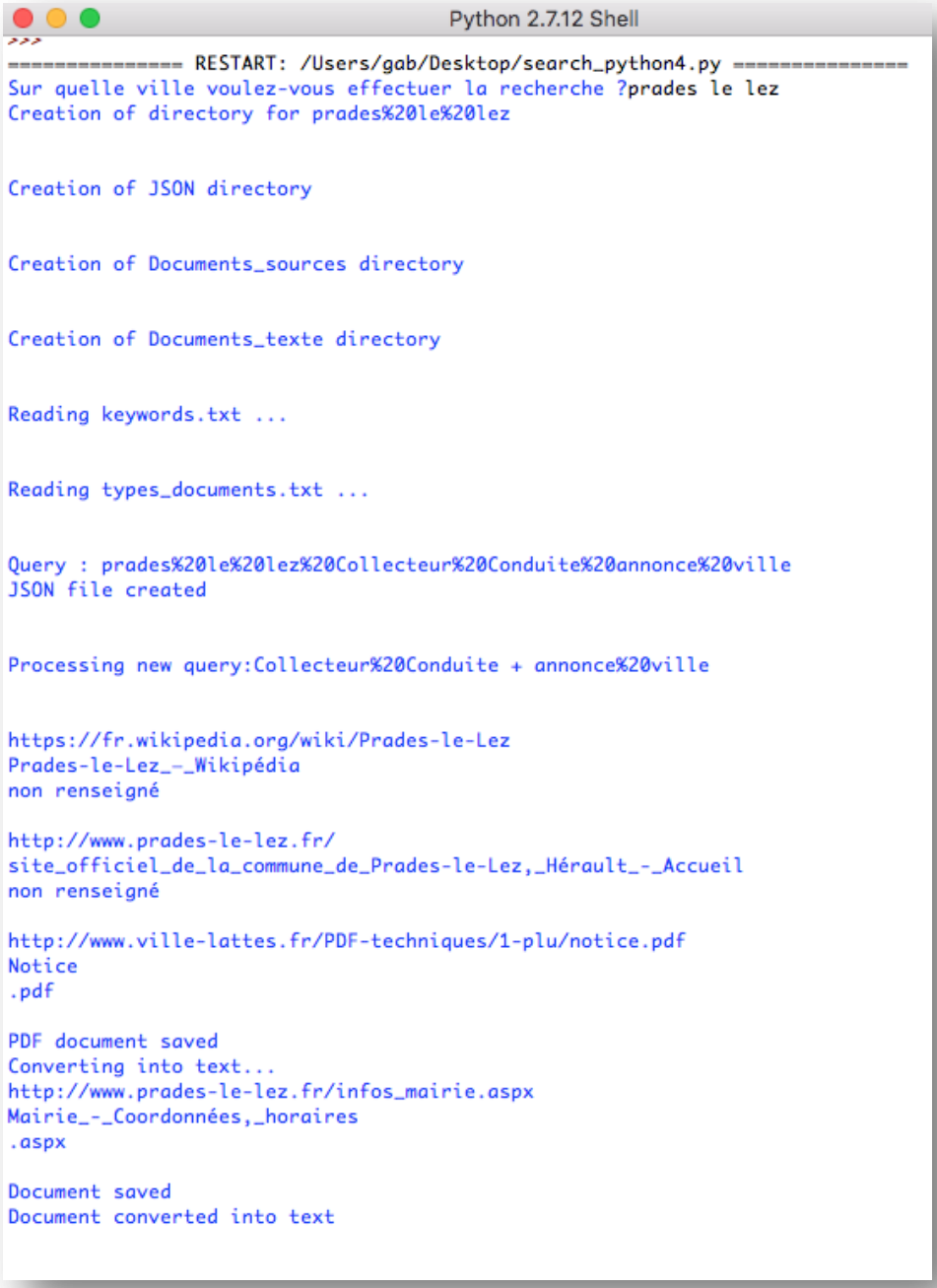


Figure 3 - Screenshot of the shell when running the program

j. Encountered difficulties and modification of the method

All along the development of this program, we encountered some difficulties and our code has undergone several modifications.

Indeed, in the beginning, the keywords were included in the python file under the form of a list. To build the queries, we chose randomly two technical keywords and one type of document. After some talks with the Cart'Eaux project members, we modified our program in order to make and process all the combinations of keywords. In agreement with them, as the

associations were too large, we chose to move them in text files that are read by the program. In this way, the user can modify the keywords if he needs to.

The execution of the program was long and we found out that some identical documents were stored each time they were retrieved. In order to reduce the time of the execution, we modified the program so as if a document is found several times with different queries, it will not be downloaded more than once, excepted if the modification dates (if they are accessible) are not the same.

We also had errors that interrupted the execution of the program. Some of them were about our quota of queries which was exceeded so we had to wait 24 hours before using that API key again but we created several Google accounts with other API keys so that we were not bothered with this error. Some errors were about the network connection which was stopped, during the execution of the program (external error due to some bad internet connection). However, an error was more difficult to solve. It concerns access rights and certification. This error was coming out each time the downloading of a PDF document was blocked and this was interrupting the execution of the program. In order to avoid this error, the document is not saved and the program continues to run. The problematic document can still be found in the report file.

Our most important difficulty was to test our solution in real conditions. Actually, when the program was running, it was not possible to run another program in parallel. At the beginning, as we needed to test the solution each time a new functionality was implemented, we started to execute the program on few queries in order to not waste resources and time. Then, once we had implemented the pause of the program, we tested it progressively on increasing period of time. At the beginning, we checked if the program was able to stop and restart with a pause of 10 seconds, then with 1 hour and finally during 24 hours.

k. Conclusion

Our program is functional and it is possible to access documents that have been stored before the break.

Benjamin COMMANDRE carried out the text analysis on documents that we have sent him. On the 81 documents collected with queries on Prades-le-Lez:

- 71.6% contain at least a city of the Montpellier urban conglomeration
- 92.5% contain at least a technical keyword
- 66.6% contain at least a city of the Montpellier urban conglomeration **AND** at least a technical keyword.

Thanks to Nanée CHAHINIAN's analysis, in this 66.6% (that represent 54 documents), we have 3 duplications. If we consider the 51 documents that are unique, we have:

- 24 documents that are not relevant (47.1%);
- 14 documents that are a little relevant (27.5%);
- 4 documents that are relevant (7.8%);

- 9 documents that are highly relevant (17.6%);

Here, we can calculate the precision of the search. The precision is the fraction of retrieved instances that are relevant.

$$Precision = \frac{tp}{tp+fp} = 25.5\%$$

tp are the true positive. For us it corresponds to the "relevant" and "really relevant" documents. ***fp*** are the false positive, the errors, it corresponds to the "not relevant" and "a little relevant" documents.

There is another statistic named recall that is the fraction of relevant instances that are retrieved.

$$Recall = \frac{tp}{tp + fn}$$

fn are the false negative, it corresponds to the entire documents that we should find.

However, we cannot calculate this statistic, because, it is not possible to know the ***fn***.

I. Self-criticism

The execution time of our program is slow. At the beginning we did not expect to run the queries on all the combinations of keywords, but after some talks with the Cart'Eaux project members, we changed our mind and decided to do so. This brings the total execution time to about 3 months and this does not cause a problem for them. It is still possible to subscribe to the paid plan in order to reduce this time of execution.

Moreover, our program is destined to be integrated in the web application that already exists and will be paired with the word analysis program. Thus, this will constitute a complete process.

Even if we are very proud of the work accomplished, there are still some documents that are not relevant. An extensive study of the queries model could be done to improve the quality of the results. However, it is not possible to end up with 100% of relevant results. Actually, even by making queries on Google manually, there are always some results that are not related to the topic searched by the user because of synonyms or because some keywords are missing.

Finally, the program we have developed can be used for any other subject. The user has just to change the keywords in the text document.

V. NEWS MONITORING SOLUTION

The Cart'Eaux project members wanted a monitoring tool that would allow them to monitor periodically town council websites in order to be informed when a news update, related to the project topic, comes out.

This second mission involves studying the feasibility of such tool and suggesting solutions to fulfil their needs.

a. Programmable solution

The first solution which we thought about was to reuse our collection of web documents program. Indeed, by adapting the queries to news search, we thought that retrieving the last news published online might be possible. Unfortunately, by running queries on Google, we can only access to the homepage of city news without accessing a particular piece of news because they do not appear in Google search results.

Another solution could be to develop a program which would have a list of city websites to visit periodically. Nevertheless, several issues come out.

The news related to a city can be found in the section "Actualités" from town council websites. The latter is sometimes reachable directly from the website's homepage, or via a tab. The topics of this news are often very varied: they concern events (exposition, cinema, market, etc.) that will or have happened, city council assessments, construction and public works notices, etc. However sometimes, on the same city website, news can be scattered in several sections. For example, for the Arles city website, we can find news in the section "Arles Info", in the calendar or in the municipal magazine downloadable in PDF format.

No city website has the same structure, the news topics are very wide and most of them do not match with the studied field, several sections from the same website can provide news. All these arguments make it impossible to carry out, as our program previously developed, an automation of the search news. The only programmable solution would be to develop a program adapted to each website, which is not conceivable in terms of efficiency and productivity.

b. RSS feed

When monitoring is the issue, RSS feeds can be an excellent tool to be warned of each update of the content of a webpage.

> *« RSS (Rich Site Summary; […] often called Really Simple Syndication) uses a family of standard web feed formats to publish frequently updated information: blog entries, news headlines, audio, video. An RSS document (called "feed", "web feed", or "channel") includes full or summarized text, and metadata, like publishing date and author's name.*
>
> *[…]*

> *Subscribing to a website RSS removes the need for the user to manually check the website for new content. Instead, their browser constantly monitors the site and informs the user of any updates. The browser can also be commanded to automatically download the new data for the user.*
>
> *Software termed "RSS reader", "aggregator", or "feed reader", which can be web-based, desktop-based, or mobile-device-based, presents RSS feed data to users. Users subscribe to feeds either by entering a feed's URI into the reader or by clicking on the browser's feed icon. The RSS reader checks the user's feeds regularly for new information and can automatically download it, if that function is enabled. The reader also provides a user interface. »* ***[7]***

The most famous feed readers are Feedly and Netvibes. They are free and offer paid functionalities.

Here, once again, the same issue arises, that is to say the difference of functioning for each city website. Indeed, very few offer RSS feeds related to their news.

Nonetheless, tools exist to overcome this lack.

c. Distill Web Monitor

This plug-in is available via the web browsers Chrome, Firefox and Opera but also on smartphones (for the moment, exclusively available for premium users). It allows monitoring a whole webpage (or just a part of it), or an RSS feed and be informed of the updates by SMS or e-mails. Some notifications show up on the browser with sound. The notification method can be configured as wanted. It is also possible to set up conditions in order to follow specific updates: when certain words appear, if new text matches a regular expression, if the text changes up to a certain percentage, etc.

The monitored webpages are organised in the form of an Inbox named “Watchlist” and updates are highlighted in green. Hereinafter you can see an example.

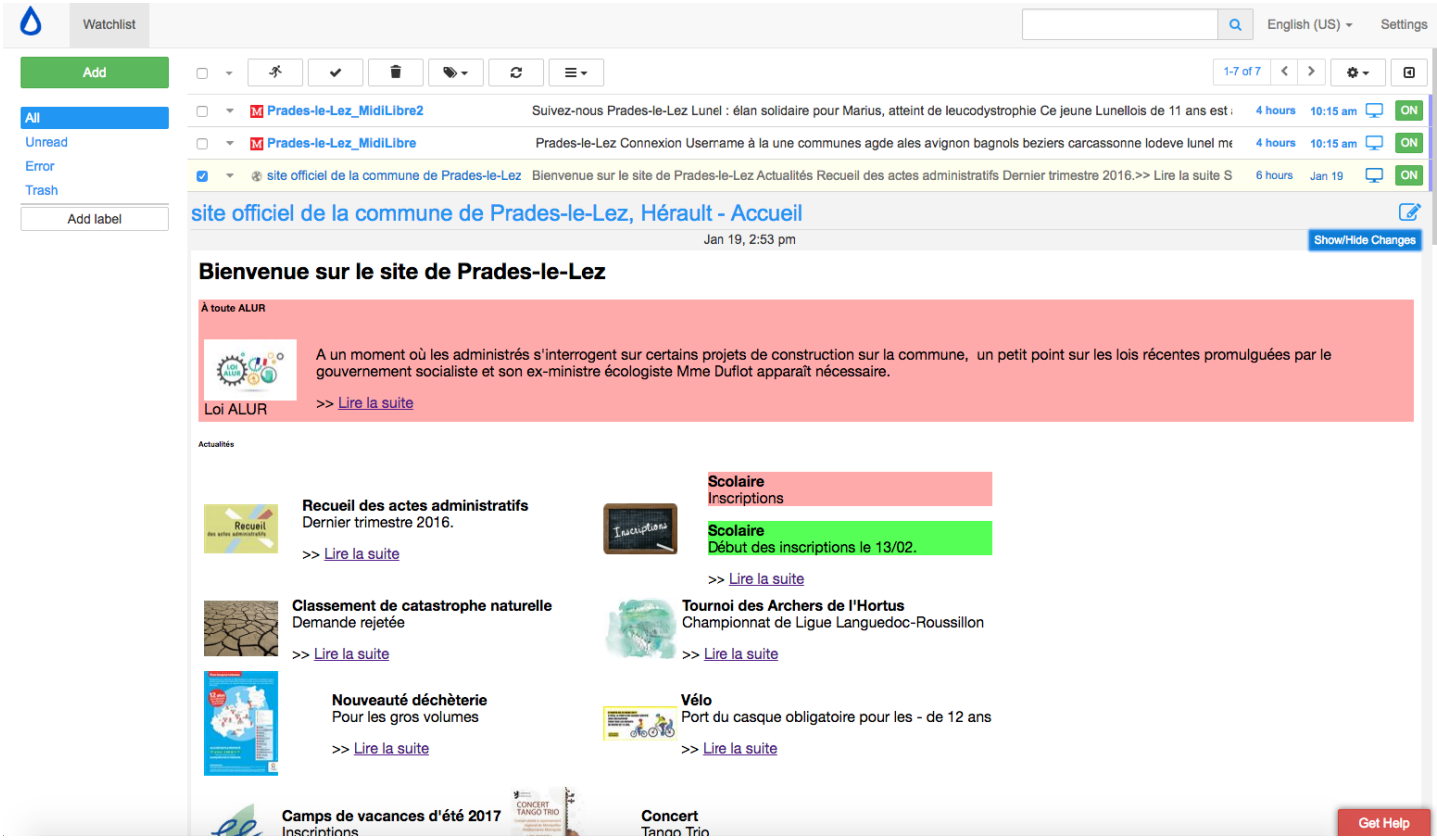


Figure 4 – Screenshot of the Watchlist

You can find a video demo of the tool functionalities using the following link:

https://www.youtube.com/watch?v=ZTMscRMFa9A

The free plan allows visiting websites periodically with order of magnitude from minutes to month. It limits the number of checks by month to 1000 and to 30 email alerts by month. The prices can be found via the following link: https://distill.io/pricing

d. Update Scanner

Update Scanner is a plug-in for Firefox which allows informing about updates on a webpage. This extension is useful for webpages on which updates are not handled by an RSS feed.

Update Scanner will automatically check the modifications of a static webpage for a given website and send an email alert.

The way it works is very simple: once you are on the webpage to monitor, you can define the frequency of the verifications and choose the importance of the modification from which you want to be warned. Hereinafter, you can find a video demo by the following link:

https://www.youtube.com/watch?v=DneeQEFSr-M

e. Conclusion

Distill Web Monitor and Update Scanner seem to be good tools to put news monitoring in place on city websites and could fulfill the expectations of the Cart'Eaux project members. Thus, they could have a global view on several websites and to be warned at each update.

Distill Web Monitor offers the advantage of being warned depending on words that appear on the website, and of selecting a part of the webpage. This avoids being warned for off-topic news or updates that do not concern news.

In addition to monitoring city websites, we thought that it could be interesting to put news monitoring in place on the regional newspaper websites as midilibre.fr which permits to search news concerning a particular city.

VI. PROJECT MANAGEMENT

As future engineers, it is essential to manage projects from the beginning until the end by way of adapted tools choice.

a. Tools used

First of all, we used Tom's Planner, a web application of project management. It allows us to create a Gantt diagram to plan our project and to keep up to-date with the progress. This graphic realization called for reflexion especially for each loads estimation tasks.

At the beginning of our project, we identified three distinct steps: the guidance of the project, the robot solution development and the news watch program development. Each of these steps assembles steps. Moreover, each week, we had to ask for validation to Carole and Nanée.

During the robot solution development, we did not use a specific tool to manage this step. We identified different tasks that we wrote on the whiteboard in our project room. As soon as a task was fulfilled, we started the next one.

b. Relations with Cart'Eaux members

Relations with Cart'Eaux project members were professional and relaxed. We met once a week and each meeting went on very well. We had the chance to be supported by Carole DELENNE and Nanée CHAHINIAN, who were attentive and available, as well as Benjamin COMMANDRE and the other members of the project.

They guided us at best and they understood the educational aspect of our work. They answered our questions very quickly and that allowed us to not waste our time on a difficulty and to keep on working.

Concerning the choice of the subject, this is when we read the subject proposed by the Cart'Eaux project members that it aroused our interest. However, after the first meeting, we realized that it was not a data-mining project but a data collection project. Even if it was not the project we imagined, we were happy to work on something that we had never studied.

To make sure our project ran smoothly, we defined fixed hours of work and we did our utmost to respect them so that we could take advantage of the time we had at our disposal.

Finally, we chose to work together on the implementation, even if we worked separately to find the best technology to use and also in case of errors, to find a solution quickly.

c. Deliverable

Concerning the deliverables, we have created a Git on which we put our summary note, our technical report and our scientific poster. We made available the source code of our solution with all the necessary documents required for the configuration and execution of our program.

Our Git is accessible via the following link: https://github.com/kidgabi/PIFE-IG5---Cart-Eaux

d. Self-criticism

We are satisfied with our project management because we have been able to overcome difficulties and to adapt our schedule depending on the progress.

It was easy to organize ourselves during the project because we worked together in the same room and we could count on each other in case of problem.

This organisation was made easier thanks to the availability of the Cart'Eaux project members, who met us weekly and quickly answered our questions.

CONCLUSION

The industrial project that we realized allowed us to acquire and apply a lot of knowledge in technical, methodological as well as interpersonal relationship.

It was very rewarding because we had to find some solutions to fulfil the needs of the Cart'Eaux project members. Our work was separated in different steps: first doing some searches in order to choose what the best solution was in terms of legality and technique. Then, presenting our results to the project members in order to have their approval on which solution to develop. These reinforced our idea that it is really important to clearly and regularly communicate so as to be sure that we all go in the same direction.

Even in the development stage, it was important to keep this dialog to make sure that the program best fits the demands of the project members.

In our opinion, the primordial step of our work was the choice of the technology we will be using all along the development stage. Indeed, by choosing the Python language, it made the development easier because we had all the tools at our disposal in order to implement the solution.

Concerning the project management, it was really hard to define the workload in advance because we had never worked on such project before. We had to adjust our schedule depending on the progress and the mishaps encountered.

Even if the ratio of relevant documents could be improved, we find that we have provided a satisfactory and well developed work of the program for the Cart'Eaux project members. Moreover, we carried out a second optional mission concerning the monitoring tools, during the remaining time.

Finally, even if the mission was not what we thought at first sight, we enjoyed working on it and it was an enriching experience that will be useful in our future professional life.

WEBOGRAPHY

[1] Cart'Eaux project presentation, application file « Projet d'avenir », Languedoc-Roussillon region, edition 2015

[2] CHAHINIAN, N., PIAT-MARCHAND, A-L., BRINGAY, S., TEISSEIRE, M., BOULOGNE, E., DERUELLE, L., DERRAS, M. and DELENNE, C.
How can big data be used to reduce uncertainty in stormwater modelling?
Proceedings of Spatial Accuracy 2016
ISBN: 978-2-9105-4510-5

[3] Wikipedia contributors, Web crawler, In *Wikipédia, The Free Encyclopedia* [Online], https://en.wikipedia.org/wiki/Web_crawler (read on 3rd December 2016)

[4] GEEK ME UP, « Comment automatiser le web (robot, crawler, scraper) ? », update on 4th September 2013, [Online] http://geekmeup.fr/comment-automatiser-le-web-robot-crawler-scraper/ (read on 2nd December 2016)

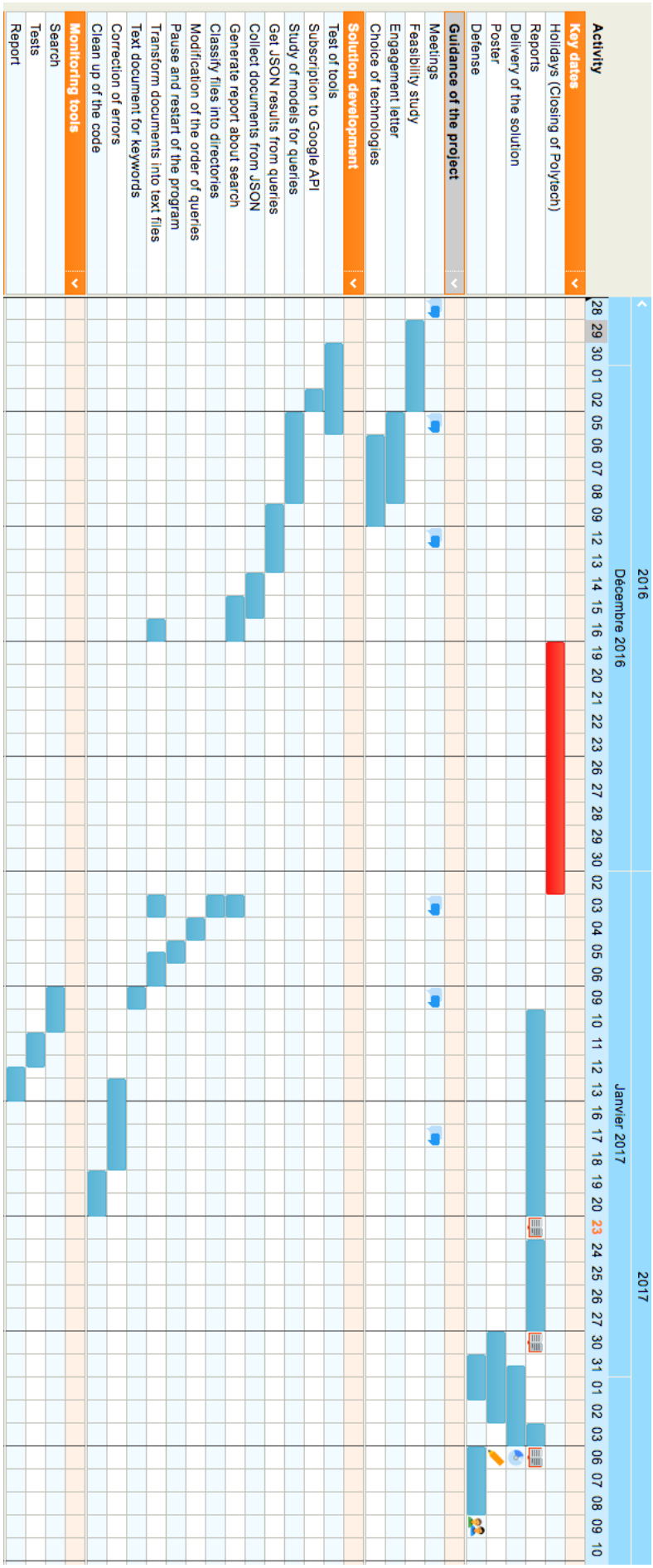
[5] Google Privacy & Terms, Google Terms of Service, update on 1st March 2012, [Online], https://www.google.com/policies/terms/archive/20070416-20120301/ (read on 9th January 2017)

[6] Google Custom Search, «Custom Search JSON/Atom API», update on 29th November 2016, [Online] https://developers.google.com/custom-search/json-api/v1/overview (read on 2nd December 2016)

[7] Wikipedia contributors, RSS, In *Wikipédia, The Free Encyclopedia* [Online], https://en.wikipedia.org/wiki/RSS (read on 10th January 2017)

APPENDIX

Appendix 1: Gantt diagram



Résumé :

Nous avons travaillé pour le projet Cart'Eaux. Il a pour but de développer une méthode de récolte de donnée multi-sources et de fusion des connaissances en une information permettant la cartographie et la modélisation hydraulique d'un réseau d'assainissement urbain.

Notre mission était d'automatiser la recherche et la collecte de documents en rapport avec les réseaux d'assainissement et les eaux pluviales pour une ville spécifique, sur le web, et de les transformer en documents texte.

Ce rapport traite également d'outils de veille d'actualités afin d'être averti dès qu'une nouvelle actualité, relative à notre sujet de recherche, apparait sur le web.

Mots clés : Projet Cart'Eaux, Python, recherche automatisée, outil de veille d'actualités

Abstract:

We worked on the Cart'Eaux project. This project aims to develop multi-sources data collection methods and merge knowledge into information that may be used to map wastewater and stormwater networks.

Our main mission was to automate the search and the collection of documents related to the stormwater and wastewater networks of a specific city on the web, and to transform them into text files.

This report also deals with monitoring tools in order to be warned when any news related to a city appears on the web.

Keywords: Cart'Eaux project, Python, automated search, monitoring tool