Fall Internship Project
Report

Develop a system that can analyse and classify PDF archaeology dataset

Submitted by

Amine Barrak
PhD Candidate
Université du Québec à Chicoutimi

Under the guidance of

Darine Amayed
École de technologie supérieure

and

Fehmi Jaafer
Université du Québec à Chicoutimi

Department of Software Engineering
Université du Québec à Chicoutimi
555 Bd de l'Université, Chicoutimi, QC G7H 2B1

Fall Internship 2021

Abstract

This project has a purpose of creating the dream archaeology of Saguenay city. The idea is to verify the city plans if they are concretely developed or under development. These plans can be related to forestry, culture, health, transport, education, etc. To do so, we need to study PDF files files that were collected from several sources such us government, organisations, cities, etc.

To explore the PDF files, we need to convert the format to textual form and then apply several natural language processing techniques to extract knowledge from these documents. From these techniques, we applied (1) keyword extraction to have an idea about the most common words that can describe the archaeology of Saguenay. (2) Topic modeling, in this part of the project we apply unsupervised algorithms to classify the documents according to their similarity.

Contents

Chapter 1

Introduction

1.1 Project Context

The purpose of this project is realised to study the dreams of the archeology Saguenay region, to see and verify if what is being said is becoming reality. The dream archeology research is based on a data collection from different sources and establish several processing to highlight ht emost important thematic.

The data proposed in this project is a set of documents that is talking about the Saguenay region in different area. These documents contain information about the region strategy, planning, reporting and policy documents. These dataset was collected and still be collected from basically four different sources: organizations, ministries and governments, MRC (County Regional Municipalities), cities and sectors of activities.

Concretely, when analyzing these documents, it should be understood and observed whether there is a concrete plan or a real transition on the part of their editors. All of this work is to draw and get an idea of the archeology dream of the Saguenay region.

The task of our research internship is to analyse the collected documents and extract the most significant words in a document *i.e.,*words frequency. After that, we aim to determine the document thematic by applying advanced NLP (Natural Language Processing) techniques. Figure 1.1 show an idea of the processing we plan to do.

1.2 Project Timeline

The project internship is set for four months, we planned the following timeline where we describe the principle to reach the different objectives.

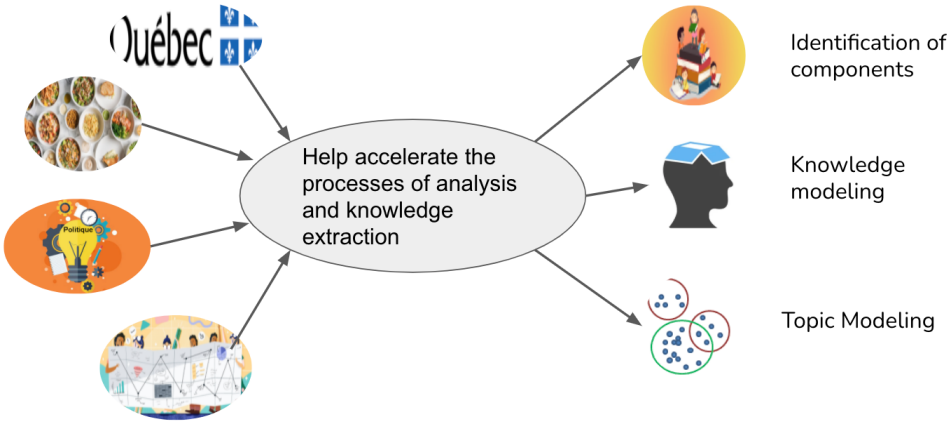


Figure 1.1: Objective and Motivation

We divided the project into a main three parts:

- PDF conversion to Text
- Keyword extraction
- Topic Modeling

For each part, we will set a documentation of the possible techniques to apply, we will develop our solution and evaluate it.

Table 1.1: Project Timeline

Topic	Detailed task	Delivery
PDF conversion to Text	**PDF organisation**	**27 sep**
	Documentation about techniques and scripts that make pdf conversion- SLR .P1	**1 oct**
	Script For PDF conversion to Text(test clean pdf)	**1 oct**
	Extend script for all pdf Pre-processing (traiting anomaly in pdf conversion Save results in CSV or database)	**29 oct**
Keyword extraction	**Documentation about models keyword extraction**	**5 nov**
	Script for Keywords Extraction	**19 Nov**
	Collaborate with Olivier's team to get Topics to link them with our topic/ keywords.	**26 Nov**
Topic Modeling	**Documentation about NLP models for topic modeling**	**3 DEC**
	Thematic and topic extraction (adjustment) Script for topic modeling	**10 DEC 17 DEC**
	Evaluate the unsupervised models with golden dataset (Olivier Team)	**7 JAN**

Chapter 2

Project Design Study

In this chapter, we will explain the different details of this project that we went throw.

2.1 Methodology Overview

We show in Figure 2.1 the different steps of this internship project. First, we need to find the most appropriate technique that can identify the different elements in the PDFs *i.e.*,text, tables and images. During our internship, we will focus only on the textual element. Second, we used the textual element to identify the most important keywords in the text. Third, we applied unsupervised clustering algorithms for topic modeling to classify the textual elements in groups according to their similarity.

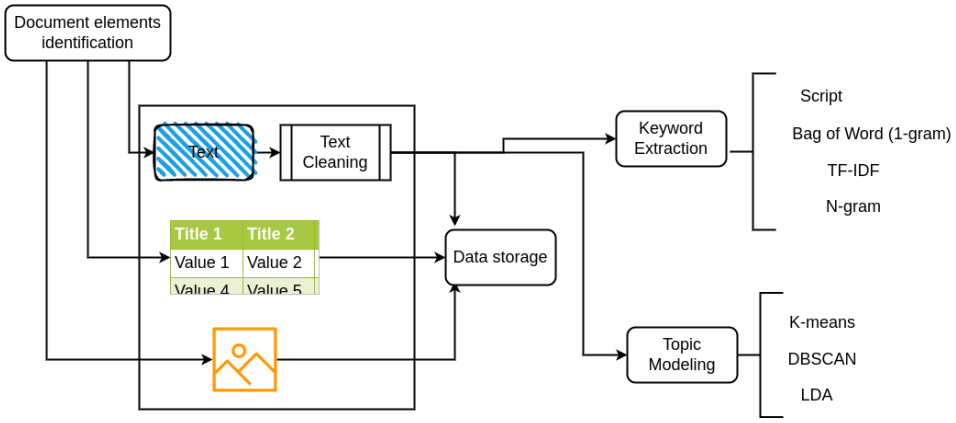


Figure 2.1: Project Methodology Overview

2.2 Dataset Presentation

The database of the project is a set of PDF files that were collected from different sources *i.e.,*government website, municipalities, cities, etc. Moreover, thematic that were searched are related to different fields, *i.e.,*community, tourism, forestry, culture, health, transport, education, etc.

These documents have no written patterns since they were provided from different sources. This make the document analysis a challenge to overcome.

2.3 The PDFs Pre-Processing and elements identification

In this project, we aim to automate an extract knowledge from the PDF files in order to attain our goals. To do so, we start by a manual PDF files classification to regroup the similar structured ones. These are the different structures:

- Presentations that were transformed to PDF: These are the files that were a presentation and transformed to PDF. Usually, the PDF pages contain unstructured small peaces of text.
- Left rotated PDF files: These are the files that were created on a landscape format.
- Structured PDF form: This is the best PDF structure, where the text is organised in one column.
- Special PDF form: These are the files that uses more than one column in their text structure.
- Corrupted PDF form: The PDF is not structured. For example, the text is split in two different pages.

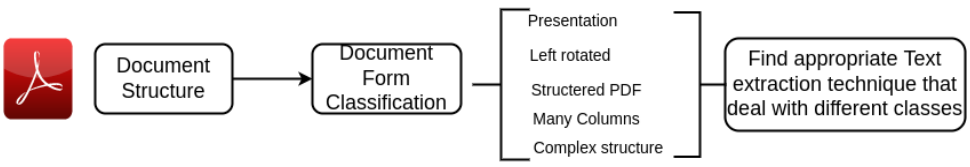


Figure 2.2: The manual PDFs structure classification

We identified 3 elements in a PDF that we can extract, (1)images ; (2) tables; and (3)Text.

2.3.1 Image extraction

In the Pdf, we can find several images that we think they might be useful for future analysis for a better representation of the Pdf content.

To extract the image element, we used a python library called "pillow". We used this library to iterate through all the pages of PDF and get all images objects present on every page.

2.3.2 Table extraction

There is several python packages to extract tables from Pdf, we chose to try two libraries *i.e.,*camelot and tabula. In most of cases, we were receiving incorrect conversion and detection of tables from the PDFs.

We chose to drop this part of table element identification, since in the tables will be considered as text. We will consider the table content as textual element.

2.3.3 Text extraction

We plan to use python during this project, since it is a high-level, interpreted language with a relatively easy syntax, authors are more familiar with python. Popular Python libraries are well integrated and provide the solution to handle unstructured data sources like Pdf and could be used to make it more sensible and useful [1].

We start by investigating the different available packages that extract text from PDF. According to Andreou [2], the following techniques are the most used ones: PyPDF2 [3], tika [4], pdfPlumber [5], pdfMiner [6]. The selection was based according to research across StackOverflow, Reddit and google searches.

We will chose from these packages the ones that can make a clean conversion of the PDF, including the special characters *i.e.,*accent, apostrophe, etc. Moreover, we need to rotate a PDF page to make the conversion properly from the left to the right. Ideally, we examine the PDF page per page and determine if a rotation is required to avoid rotating all the PDF according to its first page.

In the following we present the most discussed and used python packages for PDF conversion.

- PyPDF2 [3]: it is characterised by extracting metadata and broke the result into multiple lines, splitting documents, combining files, rotating pages, watermark.

- Tika [4]: It can be used with over 1400 file types and can deal with several languages special characters.
- PdfPlumber [5]: It primarily focuses on parsing PDFs, analyzing PDF layouts and object positioning, and extracting text.
- PdfMiner [6]: It is a tool for extracting information from PDF documents. It is used for text analysis: location, fonts, lines, etc.

There is several criteria we can use to compare between these packages, such us the fastest [7], popularity, etc. In our case, we run an experimentation to compare the effectiveness of these packages. The evaluation will be set according to the conversion quality.

We will start by converting the first page of the pdf entitled "ApprocheDD2 Résolu 2020.pdf". Figure 2.3 show the pdf page that we will convert. We applied the four packages cited above, and we. got the following result in Figure 2.4.

Rapports sur le développement durable

Approche en matière de communication du développement durable

Depuis 2008, Résolu surveille un éventail d'indicateurs de performance environnementale, sociale et économique dont elle fait état et qui sont validés à l'externe. Nos rapports sur le développement durable sont publiés en ligne aux fins de consultation publique, dont quatre rapports dans la section Publications de notre site Web, préparés selon les lignes directrices de la Global Reporting Initiative (GRI) pour 2010, 2011, 2012 et 2016, ainsi que le rapport 2018, conforme aux normes de la GRI.

Après une analyse exhaustive des besoins des parties prenantes et des pratiques exemplaires de l'industrie, nous avons remanié la section sur le développement durable de notre site Web pour y inclure des données détaillées et des références connexes.

Cadre de la GRI pour la production de rapports de développement durable

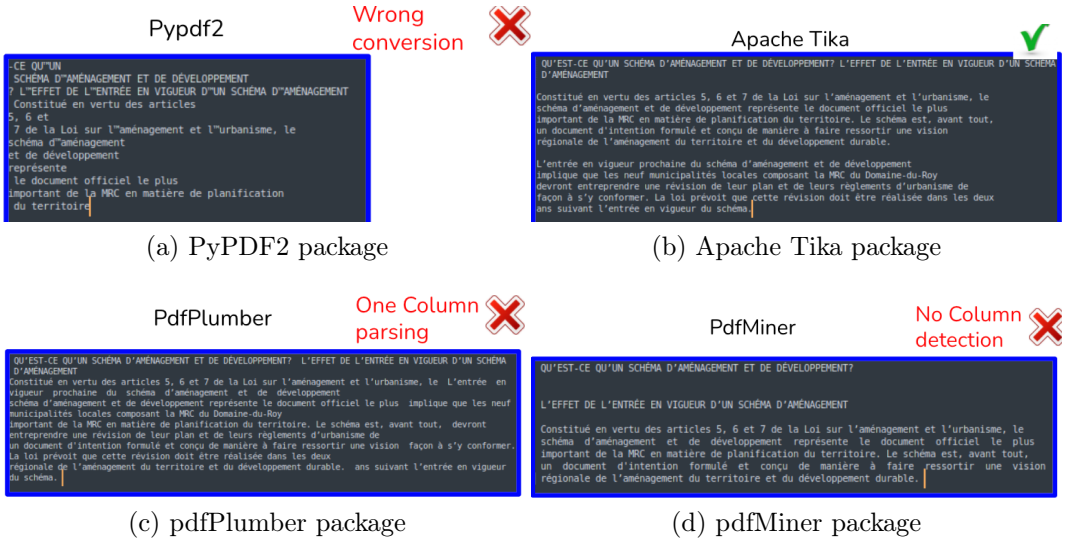
Les rapports de développement durable qui sont publiés sur notre site Web et l'index GRI connexe ont été préparés en conformité avec les normes de la Global Reporting Initiative. Nos rapports respectent les normes de la GRI : option de conformité essentielle, ce qui comprend l'information générale requise ainsi que des données de gestion génériques et des indicateurs liés à chacun des enjeux pertinents (c.-à-d. nos priorités communes).

La publication sur notre site Web des rapports de développement durable conformes aux normes de la GRI nous permet de mettre l'information à jour plus facilement et de la rendre accessible à l'ensemble de nos parties prenantes. De plus, comme un grand nombre de nos pairs, nous avons adopté un cycle de production de rapports semestriels. Nous produirons et déposerons tous les deux ans un index GRI fournissant toutes les données environnementales, sociales et économiques pertinentes pour les trois exercices précédents.

Figure 2.3: The PDF page used to test the PDF converter package

According to the results, as it was expected we can clearly see that Apache Tika has the best accurate results. We identified that (1) PdfPlumber do not take in consideration multiple columns parsing, it parse the pdf text line by line without considering the the separated paragraphs; (2) Pypdf2 make a wrong text conversion, it do not convert correctly the special characters *i.e.*,, apostrophe; (3) PdfMiner do not keep order of the paragraphs especially in cases of multiple columns in a pdf pages. We consider apache Tika, the best tool to extract text from PDFs.

For more complicated cases where the PDF is left rotated (landscape form), presentation or multiple columns in the PDF. We found a difficulty to parse their pages, since all tools were not designed to parse such cases.



(a) PyPDF2 package (b) Apache Tika package

(c) pdfPlumber package (d) pdfMiner package

Figure 2.4: Python packages for textual extraction from PDF

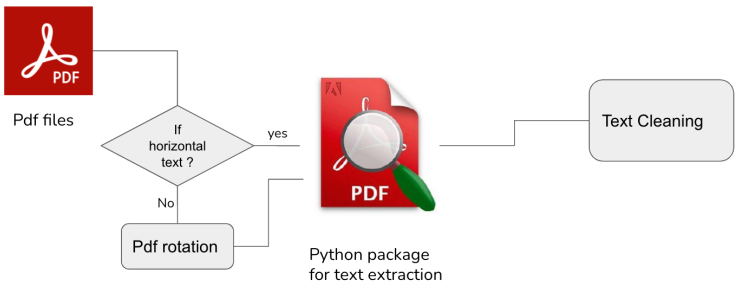


Figure 2.5: PDF rotation

We found that it is necessary to rotate the pdf pages in order to have a clean extraction. When we applied rotation to the PDF, we had an acceptable result. However, there is some case where parts of a page in a PDF are rotated *e.g.*,text in a table. These parts of the text will be split in several lines, where each line contain one or two characters. This problem can be fixed in the pre-processing text cleaning.

We chose to rotate manually all the PDFs categories to make the text horizontal as shown in Figure 2.5, then, the script can easily convert the pdf content to text. We used the online tool pdf2go [8] to apply the pdf

```python
def rotate(pdf_path_imput, pdf_path_output, list_pages):
    pdfIn = open(pdf_path_imput, 'rb')
    pdfReader = PyPDF2.PdfFileReader(pdfIn)
    pdfWriter = PyPDF2.PdfFileWriter()
    for pageNum in range(pdfReader.numPages):
        page = pdfReader.getPage(pageNum)
        if pageNum in list_pages:
            page.rotateClockwise(90)
        pdfWriter.addPage(page)

    pdfOut = open(pdf_path_output, 'wb')
    pdfWriter.write(pdfOut)
    pdfOut.close()
    pdfIn.close()
```

Figure 2.6: PDF rotation script

pages rotation. We created also a script based on Pypdf2 to rotate the pdf pages, where we specify the page number that we need to rotate as shown in Figure 2.6. There is other possibilities to rotate the pdf pages such as PyOCR package [9] that can detect the angle of an image page, there migh be a future usage of such package to automate the pdf rotation before the text extraction.

2.3.4 Estimation of the rotated text

We want to make an estimation of the rotated words in our documents. We count the number of characters in lines where text extracted was rotated. Since, the french average word length is: 5.13 Characters [10], we estimated the percentage of the text rotated is about 0.29%.

We processed to clean these rotated parts of the documents and fix the lines where the words are probably incorrect due to the text extraction from the PDFs. First, we searched for a french 1-gram dictionary and we found an open source dataset presented by Google [11]. Second, we applied an algorithm OneGramDist [12]. It consist on parsing every line of our textual dataset and it tries to choose the most likely spelling correction for each word based on the dictionary we provide. The algorithm is costly in term of processing and time of execution, since a line can have a long paragraph and it had to search for the right characters distribution in each word.

The result was far away from a correct text, probably because of the provided dictionary is different from the vocabulary of the documents we are studying.

2.3.5 Pre-processing and text cleaning

We proceeded to clean the resulted text from the usage of the Tika tool. We applied a set of rules that are based on regular expression to eliminate the parts of text that we desire to remove.

The following rules were applied as functions:

- Space and empty lines removal

```
def remove_empty_lines(text):
        text=re.sub(r'\n(-|\.)[[:space:]]*\d+',  r'', text)
        text=re.sub(r'\n[[:space:]]*\d+',  r'', text)
        return text
```

- Remove wrong converted characters

```
def remove_weird_characters(text):
        text=re.sub(r'.*?.*\n',  r'', text)
        return text
```

- Fix non clean rotated text

```
def fix_rotated_text(text):
        text=re.sub(r'\n(?|(au)|(et)|(de)|(des))\
        ([[[:space:]][a-zÀ-ÿ]{2,4}]?)\n',\
        r' \1\2 ', text)
        text=re.sub(r'\n([a-zÀ-ÿ]{1})\n',  r'\1 ', text)
        return text
```

- Link divided paragraphs

```
def link_lines(text):
        text=re.sub(r'\n([a-zÀ-ÿ])',  r' \1', text)
        return text
```

- Link hyphen separated words

```
def link_dashed_word(text):
        text=re.sub(r'(\w)\n- (\w)',  r'\1\2', text)
        text=re.sub(r'(\w)- (\w)',  r'\1\2', text)
        return text
```

- Remove URLs

```
def remove_urls(text):
        text = re.sub(r"http\S+", "", text)
        return text
```

2.4 Dataset cloud storage

Figure 2.7: The PDF page used to test the PDF converter package

After cleaning the extracted dataset, we plan to store the different extracted elements for a future usage. We chose a non relational database, since, we will need to store images and text elements inside the database for a future processing. We chose to use MongoDB since it support NoSQL databases where we can save elements like imageswith the appropriate source files. It has also flexible schema that makes it easy to evolve and store data in a way that is easy for programmers to work with. MongoDB is also built to scale up quickly.

We prepare a docker image that contain the different required python packages for an ease share of the database. We also saved the extracted elements locally as shown in Figure 2.7.

2.5 Keyword Extraction

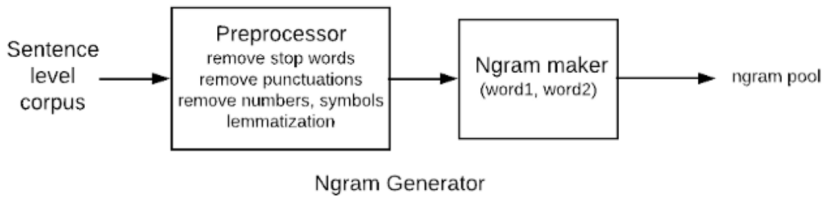
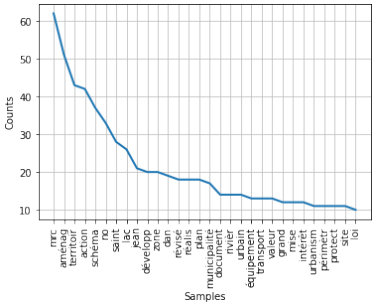


Figure 2.8: Overview of the processing to generate ngram pool

The size of documents we are facing is large with a median of value of **251328 words per document**. The Keywords or key-phrases extraction methods can be of great to have an idea about the most common words in our documents.

From the most known algorithms to extract keywords or key-phrases is the N-gram method, which means a sequence of N words. So for example, “Medium blog” is a 2-gram (a bigram), “A Medium blog post” is a 4-gram, and “Write on Medium” is a 3-gram (trigram).

A preprocessor is applied to clean and standardize the keyword phrases before the Ngram generator. This preprocessor removes stop words, punctuations, numbers, and symbols, and lemmatizer the words. After getting the clean sentence level corpus, the n-grams are all possible adjacent n-words. Figure 2.8 shows the processing to create an n-gram pool.



(a) Number of words occurences in the document

(b) Cloud words of the most occured words in the document

We applied a 1-gram processing algorithm on one file to show a sample of results. Figure 3.1b shows the words occurrences in the file and Figure 3.1a shows the words cloud of the most appeared words in the document.

2.6 Unsupervised Clustering & Topic Modeling

Unsupervised Clustering is the most commonly used unsupervised learning method. This is because typically it is one of the best ways to explore and find out more about data visually. It allows is a way to group a set of data points in a way that similar data points are grouped together. Therefore, clustering algorithms look for similarities or dissimilarities among data points. Clustering is an unsupervised learning method so there is no label associated with data points. The algorithm tries to find the underlying structure of the data [13]. We present in Figure 2.9 an overview of the unsupervised learning methods.

There are different approaches and algorithms to perform clustering tasks:

- Partition-based clustering: E.g. k-means
- Density-based clustering: E.g. DBSCAN

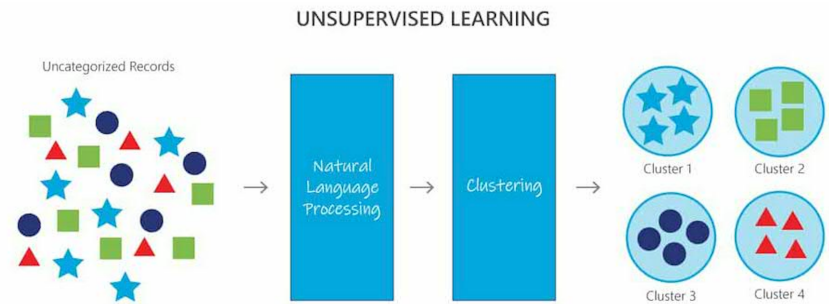


Figure 2.9: Overview of the unsupervised Learning methods

Moreover in the following, we applied Latent Dirichlet Allocation (LDA), where the assignment of each document is made according to the probability distribution of groupings—or topics—for each document. In other words, each document will be assigned to distribution of topics, with each topic having a probability associated with it [14].

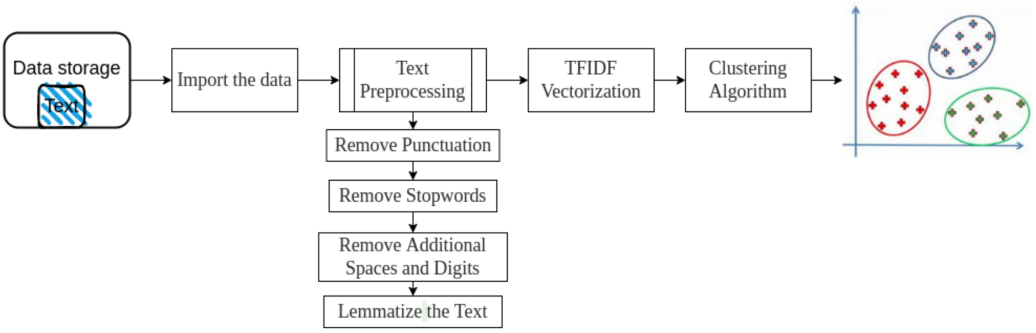


Figure 2.10: Overview of the different steps to perform the unsupervised clustering method

There is several necessary steps to made before applying the clustering algorithm, they are described as following:

- Import the data (all the docuemnts in pandas or csv)
- Text Preprocessing — cleaning the data.
- Vectorization (creating a vector of words called vocabulary).
- Clustering algorithm (K-means, DBscan, etc)
- Show clustering results

Figure 2.10 shows the different steps we applied to perform the unsupervised clustering algorithms.

2.6.1 K-means algorithm

K-means is a clustering algorithm that is known to produce generally good results. Taking in parameter k, the amount of wanted clusters, it randomly chooses k centroids from the data. For n iterations, or until convergence, move the points to the cluster where its centroid is the closest to the point. Then recompute the centroids as the mean of all points in the cluster. Convergence occurs when no points move from a cluster to another during an iteration. The K-means algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster. The 'means' in the K-means refers to averaging of the data; that is, finding the centroid [15].

To process the learning data, the K-means algorithm process with the following steps:

- Consider the decided number of clusters K (specified by user).
- Select random K points or centroids. (It can be other from the input dataset).
- Assign each data point to their closest centroid, which will form the predefined K clusters.
- Calculate the variance and place a new centroid of each cluster.
- Repeat assigning each datapoint to the new closest centroid of each cluster.
- If any reassignment occurs, recalculate the variance.

The approach kmeans follows to solve the problem is called Expectation-Maximization. The E-step is assigning the data points to the closest cluster. The M-step is computing the centroid of each cluster. Below is a break down of how we can solve it mathematically[16]. The objective function is :

$$J = \sum_{i=1}^{m} \sum_{k=1}^{k} w_{ik} \left\| x^i - \mu_k \right\|^2 \tag{2.1}$$

where:

- w ik=1 for data point x i if it belongs to cluster k;
- otherwise, w ik=0

2.6.2 DBscan algorithm

DBSCAN stands for density-based spatial clustering of applications with noise. It is able to find arbitrary shaped clusters and clusters with noise (i.e. outliers). IT is a partitional clustering algorithm. It can provide some better results than other clustering algorithms when data shows complex structures that can't be linearly separated, but still provides points close to each other when they are the same class. DBSCAN can also detect noise or outliers from data. The main idea behind DBSCAN is that a point belongs to a cluster if it is close to many points from that cluster. [17]

- Classify the points.
- Discard noise.
- Assign cluster to a core point (centroid).
- Consider all the density connected points of a core point.
- Consider boundary points according to the nearest core point.

DBSCAN is very effective in noise elimination. it is capable to detect data point that is neither a core point nor a boundary point, then it detect it as a noise point. The algorithm consider a core point when it satisfy the condition of neighbors that must be greater than or equal to threshold (number of centroid) [18].

The main idea behind DBSCAN is that a point belongs to a cluster if it is close to many points from that cluster. There are two key parameters of DBSCAN [19]:

- eps: The distance that specifies the neighborhoods. Two points are considered to be neighbors if the distance between them are less than or equal to eps.
- minPts: Minimum number of data points to define a cluster.

Based on these two parameters, points are classified as core point, border point, or outlier:

- Core point: A point is a core point if there are at least minPts number of points (including the point itself) in its surrounding area with radius eps.
- Border point: A point is a border point if it is reachable from a core point and there are less than minPts number of points within its surrounding area.

- Outlier: A point is an outlier if it is not a core point and not reachable from any core points.

2.6.3 Latent Dirichlet Allocation (LDA) algorithm

In natural language processing, the latent Dirichlet allocation (LDA) is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. Concretely , **LDA is an algorithm for topic modeling**, where it capable of applying unsupervised classification of documents. It finds some natural groups of items (topics) even when we're not sure what we're looking for. Figue 3.3 show the principle of LDA algoritm [20]:

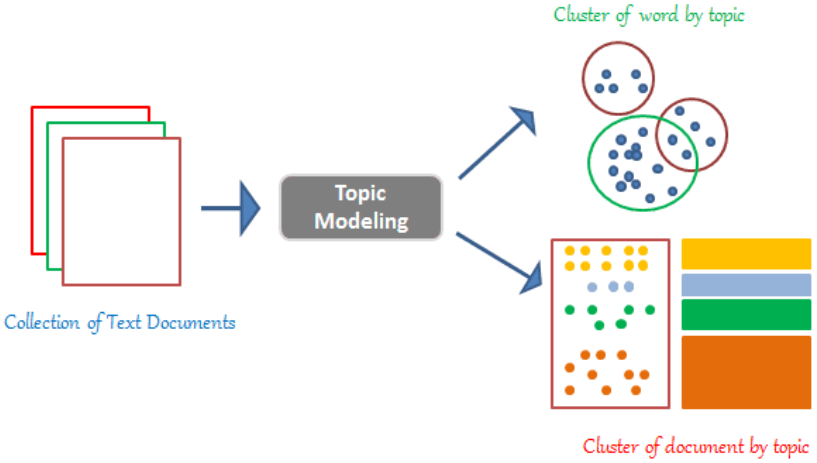


Figure 2.11: Overview of the Topic Modeling with LDA

For example, if observations are words collected into documents, it posits that each document is a mixture of a small number of topics and that each word's presence is attributable to one of the document's topics. LDA is an example of a topic model and belongs to the machine learning field and in a wider sense to the artificial intelligence field [21].

There are 2 parts in LDA [22]:

- The words that belong to a document, that we already know.
- The words that belong to a topic or the probability of words belonging into a topic, that we need to calculate.

Concretely the algorithm works with the following principle [23]:

- Go through each document and randomly assign each word in the document to one of k topics (k is chosen beforehand).
- For each document d, go through each word w and compute :
    1. p(topic t | document d): the proportion of words in document d that are assigned to topic t. Tries to capture how many words belong to the topic t for a given document d. Excluding the current word. If a lot of words from d belongs to t, it is more probable that word w belongs to t.
    2. p(word w| topic t): the proportion of assignments to topic t over all documents that come from this word w. Tries to capture how many documents are in topic t because of word w. LDA represents documents as a mixture of topics. Similarly, a topic is a mixture of words. If a word has high probability of being in a topic, all the documents having w will be more strongly associated with t as well. Similarly, if w is not very probable to be in t, the documents which contain the w will be having very low probability of being in t, because rest of the words in d will belong to some other topic and hence d will have a higher probability for those topic. So even if w gets added to t, it won't be bringing many such documents to t.
- Update the probability for the word w belonging to topic t, as

$$p(wordwwithtopict) = p(topict|documentd) * p(wordw|topict) \quad (2.2)$$

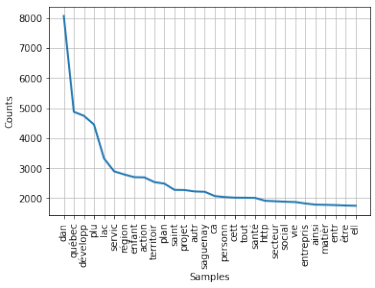
Chapter 3

Results

In this chapter, we will present the different results that contributes to reach the project goals.

3.1 Keywords extractions

We applied the 1-gram alorithm in all the documents corpus. First, we cleaned the text by removing stop words, punctuations, etc. Second, we applied a words tokenisation and lemmatisation. Then, we make sure all the words are in lowercase. Finally, we count the occurence of each word to show the most appeared ones. Figure 3.1a show the most appeared words in the documents corpus and Figure 3.1b shows the words cloud of the most appeared words.



(a) Number of words occurences in the document

(b) Cloud words of the most occured words in the document

3.2 Unsupervised clustering

In this section, we present the results of the unsupervised clustering algorithms, we will see different results of clustering according to the algorithm functionalities.

3.2.1 K-means algorithm

K-means looks for a fixed number (k) of clusters in a dataset. The K-means algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster. During this experience, we fixed the number of cluster to three. Figure 3.1 shows the results of the different clusters and the most appeared words in each cluster.

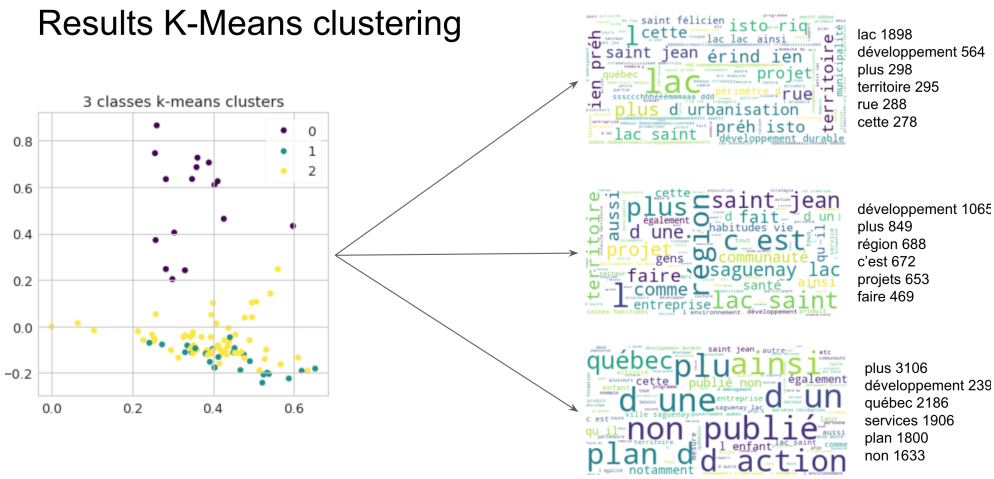


Figure 3.1: Overview of the unsupervised clustering K-means algorithm

3.2.2 DBscan algorithm

DBSCAN does not require to specify number of clusters beforehand. DBSCAN stands for density-based spatial clustering of applications with noise. It is able to find arbitrary shaped clusters and clusters with noise (i.e. outliers). During this experience, we tried to change the parameter to make the algorithm realise only three cluster so we can have an idea with the other unsupervised clustering algorithms. Figure 3.2 shows the results of the different clusters and the most appeared words in each cluster.

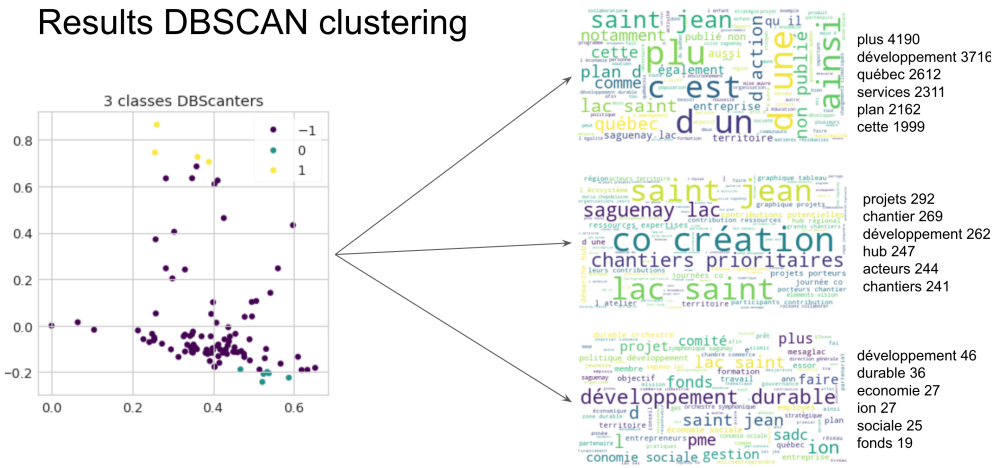


Figure 3.2: Overview of the unsupervised clustering of the DBscan algorithm

3.2.3 Latent Dirichlet Allocation (LDA)

LDA is a probabilistic model where each topic is considered as a mixture of words and each document is considered as a mixture of topics. Using LDA, we will attempt to identify the latent topics from corpus. During this experience, we have trained the model and we mentioned a three topics.

We evaluated the model with Topic Coherence measures. It is a generated for each topic by measuring the degree of semantic similarity between high scoring words in the topic. These measurements help distinguish between topics that are semantically interpretable topics and topics that are artifacts of statistical inference.

The measure is based on a sliding window, one-set segmentation of the top words and an indirect confirmation measure that uses normalized point-wise mutual information (NPMI) and the cosine similarity. **We found a Coherence Score of: 0.293**.

Then, we used a popular visualization package called "pyLDAvis" which is designed to help interactively with: Better understanding and interpreting individual topics, and Better understanding the relationships between the topics.

Figure 3.3 shows an idea about the results that the package "pyLDAvis" visualise.

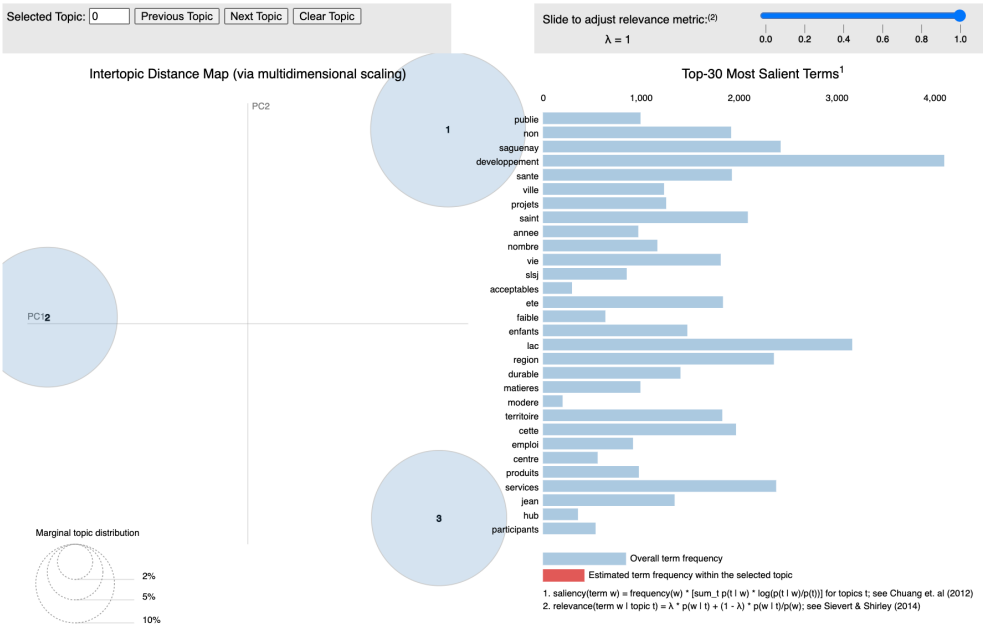


Figure 3.3: Overview of the Topic modeling algorithm LDA

Chapter 4

Conclusion

During this project, we learnt how to extract the necessary information from documents. There was several challenges that we overcome most of them. First, we chose the best python package to convert the pdf to textual form(apache tika). Second, we applied 1-gram algorithm to extract the most important keywords in our documents corpus. Finally, we applied unsupervised clustering algorithms (k-means, DBscan and LDA) to group the documents according to their similarities.

In the Future, we plan to improve the French stop-words for a better cleaning of the keywords which could helps the unsupervised clustering algorithms to make better decisions.

References

[1] "Pdf processing with python. the way to extract text from your pdf... | by ahmed khemiri | towards data science," https://towardsdatascience.com/pdf-preprocessing-with-python-19829752af9f, (Accessed on 12/12/2021).

[2] C. Andreou, "How to extract text from pdf. learn to use python to extract text... | towards data science," https://towardsdatascience.com/how-to-extract-text-from-pdf-245482a96de7, (Accessed on 12/12/2021).

[3] "Github - mstamy2/pypdf2: A utility to read and write pdfs with python," https://github.com/mstamy2/PyPDF2, (Accessed on 10/01/2021).

[4] "Apache tika – apache tika," https://tika.apache.org/, (Accessed on 10/01/2021).

[5] "Github - jsvine/pdfplumber: Plumb a pdf for detailed information about each char, rectangle, line, et cetera — and easily extract text and tables." https://github.com/jsvine/pdfplumber, (Accessed on 09/30/2021).

[6] "Github - pdfminer/pdfminer.six: Community maintained fork of pdfminer - we fathom pdf," https://github.com/pdfminer/pdfminer.six, (Accessed on 10/01/2021).

[7] "Efficient pdfs processing with python | by maciej januszewski | softserve | medium | analytics vidhya," https://medium.com/analytics-vidhya/efficient-pdfs-processing-with-python-abffd75b1af7, (Accessed on 10/01/2021).

[8] "Rotate pdf - rotate your pdf pages online," https://www.pdf2go.com/rotate-pdf, (Accessed on 10/07/2021).

[9] "World / openpaperwork / pyocr · gitlab," https://gitlab.gnome.org/World/OpenPaperwork/pyocr, (Accessed on 10/01/2021).

[10] "Average word length - puchu.net," http://www.puchu.net/doc/Average_Word_Length, (Accessed on 01/17/2022).

[11] "Google ngrams - french : Google, inc. : Free download, borrow, and streaming : Internet archive," https://archive.org/details/google_ngrams-french, (Accessed on 01/17/2022).

[12] "How to write a spelling corrector," http://norvig.com/spell-correct.html, (Accessed on 01/17/2022).

[13] "Dbscan clustering — explained. detailed theorotical explanation and... | by soner yıldırım | towards data science," https://towardsdatascience.com/dbscan-clustering-explained-97556a2ad556, (Accessed on 12/28/2021).

[14] "Is latent dirichlet allocation (lda) a clustering algorithm? - hds," https://highdemandskills.com/lda-clustering/, (Accessed on 12/28/2021).

[15] "Unsupervised-text-clustering using natural language processing(nlp) | by rohith ramesh | medium," https://medium.com/@rohithramesh1991/unsupervised-text-clustering-using-natural-language-processing-nlp-1a8bc18b048d, (Accessed on 12/28/2021).

[16] "K-means clustering: Algorithm, applications, evaluation methods, and drawbacks | by imad dabbura | towards data science," https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48 (Accessed on 01/28/2022).

[17] "Dbscan - wikipedia," https://en.wikipedia.org/wiki/DBSCAN, (Accessed on 12/28/2021).

[18] "Dbscan algorithm | understand the dbscan clustering algorithm," https://www.analyticsvidhya.com/blog/2021/06/understand-the-dbscan-clustering-algorithm/, (Accessed on 01/19/2022).

[19] "Dbscan clustering — explained. detailed theorotical explanation and... | by soner yıldırım | towards data science," https://towardsdatascience.com/dbscan-clustering-explained-97556a2ad556, (Accessed on 01/28/2022).

[20] "Topic modelling with lda -a hands-on introduction - analytics vidhya," https://www.analyticsvidhya.com/blog/2021/07/

topic-modelling-with-lda-a-hands-on-introduction/, (Accessed on 01/28/2022).

[21] "Latent dirichlet allocation - wikipedia," https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation, (Accessed on 12/28/2021).

[22] "A beginner's guide to latent dirichlet allocation(lda) | by ria kulshrestha | towards data science," https://towardsdatascience.com/latent-dirichlet-allocation-lda-9d1cd064ffa2, (Accessed on 01/19/2022).

[23] "A beginner's guide to latent dirichlet allocation(lda) | by ria kulshrestha | towards data science," https://towardsdatascience.com/latent-dirichlet-allocation-lda-9d1cd064ffa2, (Accessed on 01/28/2022).