CentraleSupélec

université PARIS-SACLAY

Big Data Research Project

Shubham BISWAS
Maria TERRAH

Text Extraction from Geological Documents

Advisors: Nacéra SEGHOUANI - CentraleSupelec nacera.seghouani@centralesupelec.fr
Francesca BUGIOTTI - CentraleSupelec francesca.bugiotti@centralesupelec.fr
Sylvain WLODARCZYK - Shlumberger swlodarczyk@slb.com

CentraleSupélec

Contents

Chapter 1

General Introduction

Nowadays much of the information is published in the form of PDF documents and many organizations , industries and companies seek to retrieve and utilize information extracted from these documents. As a matter of fact, there is a huge demand for storing information to a computer storage disk from the data available in printed or handwritten documents to later re-utilize this information by means of computers. One simple way to store information to computer system from these paper documents could be to first scan the documents and then store them as image files.
Optical character recognition (OCR) is an active research area that attempts to respond to this need by developing computer systems with the ability to extract and process text from images automatically. OCR method has been used in converting printed text into editable text.The accuracy of OCR can be dependent on text preprocessing and segmentation algorithms.

The main challenges of OCR systems is the quality of the input : sometimes it is difficult to retrieve text from the image because of different size, style, orientation, complex background of image etc. For good quality and high accuracy character recognition, OCR techniques expect high quality or high resolution images with some basic structural properties such as high differentiating text and background. The way images are generated is an important and determining factor in the accuracy and success of OCR, since this often affects the quality of images dramatically. Usually OCR with images produced by scanners gives high accuracy and good performance. In contrast, images produced by cameras usually are not as good of an input as scanned images to be used for OCR due to the environmental or camera related factors and as a result numerous errors might emerge.

Our project aims to develop a tool to efficiently retrieve text from scanned documents. The developed approach is based on three main steps : The pre-processing of the input including pre-processing of the image and table detection , text correction and finally the topic analysis process.
The present report is structured as follows : starting with an overview of the project , followed by image preprocessing and data filtering work on the third chapter. The fourth chapter is about the post-processing work including text correction and topic analysis. The fifth chapter contains the experiments and results.

The github repository to our project is accessible here [1]

Chapter 2

Overview of the Project

2.1 General context

Schlumberger is the world's leading provider of technology for reservoir characterization, drilling, production, and processing to the oil and gas industry. Schlumberger is incorporating different artificial intelligence technologies to find solution of the above mentioned work. For example, usage of machine learning methods for comprehensive data management and monitoring would improve the efficiency and quality for drilling planning and operations. They supply the industry's most comprehensive range of products and services, from exploration through production, and integrated pore-to-pipeline solutions that optimize hydrocarbon recovery to deliver reservoir performance sustainably.

The goal is to automate wellbore data interpretation to (re) assess large area in few hours and to drill wells through subterranean environments they can never actually be seen. Humans can understand the contents of an image simply by looking at it. We perceive the text on the image as text and can read it. Computers do not work in the same way. They need something more concrete, organized in a way they can understand. This is where Optical Character Recognition (OCR) is used.

2.2 Related works

There are many applications of OCR in the literature. We would like to cite some of the major contributions in the prior work in the OCR field: image text extraction from natural scene images [**2**], extracting text from scanned documents [**3**]etc. The system proposed in [**3**] is to rectify the text retrieved from images captured by camera. An OCR system proposed by Thomas Deselaers et al. [**12**] is used for recognizing handwritten characters and converting these characters into digital text.
When it comes to images pre-processing, many thresholding methods have been published in the literature,, for example, Sahoo et al. compared the performance of more than 20 global thresholding algorithms using uniformly or shape measures. The comparison showed that Otsu class separability method gave best performance (Sahoo et al., 1988; Otsu, 1979). The OCR goal-directed evaluation study by Trier and Jain examined four global techniques showing that the Otsu method outperformed the other methods investigated in the study (Trier Jain, 1995). In addition, Fischer compared 15 global methods and confirmed that the Otsu method is preferred in document image processing (Fischer, 2000).
Regarding table detection , it has already been explored in the field of deep learning but these were mostly on table data extraction in contrast to our work where we wanted to ignore the table data, therefore we had to find a different approach to solve our problem. End-to-end Table detection and Tabular data extraction is an example [**14**].

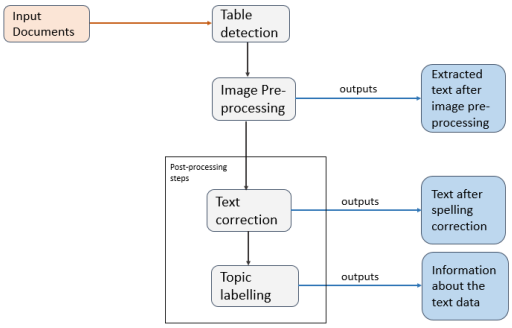


Figure 2.1: Global architecture of the solution.

2.3 Global architecture

The general architecture of our proposed solution is like shown in the figure[2.1]. The system take as input the PDF documents that are converted to images. Each page from the PDF document corresponds to an image.
The principal modules of the application and in the order are :

1. Table detection : the goal from this step is to detect and filter out images containing tables in them.

2. Image pre-processing : this module includes all the operations relate to image quality and the objective is to have images with optimized quality.

3. Text correction: this aims to correct the maximum spelling mistakes once the text is extracted.

4. Topic Analysis : the goal of this module is to gather some insights and retrieve information from the extracted text

Chapter 3

Image Preprocessing and Data Filtering

Image pre-processing is the name for operations on images at the lowest level of abstraction whose aim is an improvement of the image data. [**10**] The aim of pre-processing is an improvement of the image data that suppresses undesired distortions or enhances some image features relevant for further processing and analysis task. For our project as we are dealing with very old geological documents, image pre-processing plays an important role for image enhancement. Another constraint of our project is the need to fetch only some specific text from the document. Therefore, to accomplish this we are using some techniques to detect the unwanted pages and focus on the required text.

3.1 Importance of preprocessing for text extraction

The importance of the preprocessing stage of a character recognition system lies in its ability to remedy some of the problems that may occur during text extraction. Thus, the use of preprocessing techniques may enhance a document image preparing it for the next stage in a character recognition system. In order to achieve higher recognition rates, it is essential to have an effective preprocessing stage, therefore; using effective preprocessing algorithms makes the OCR system more robust mainly through accurate image enhancement, noise removal, image thresholding, skew detection/correction, page segmentation, character segmentation, character normalization and morphological techniques.

3.2 Preprocessing techniques

Preprocessing techniques are needed on text images particularly scanned images. . Often scanned images contain non-uniform background and/or watermarks making it difficult to extract the document text from the image without performing some kind of preprocessing.

To achieve this, several steps are needed, first, some skew correction techniques to extract characters in the right order , followed by thresholding to remove the background containing any scenes, watermarks and/or noise,and finally some image quality improvement to remove noise or correct the contrast in the image.
The above techniques present few of those which may be used in character recognition systems and in some applications; few or some of these techniques or others may be used at different stages of the OCR system. The rest of the chapter will present the techniques used during the preprocessing stage of our character recognition system.

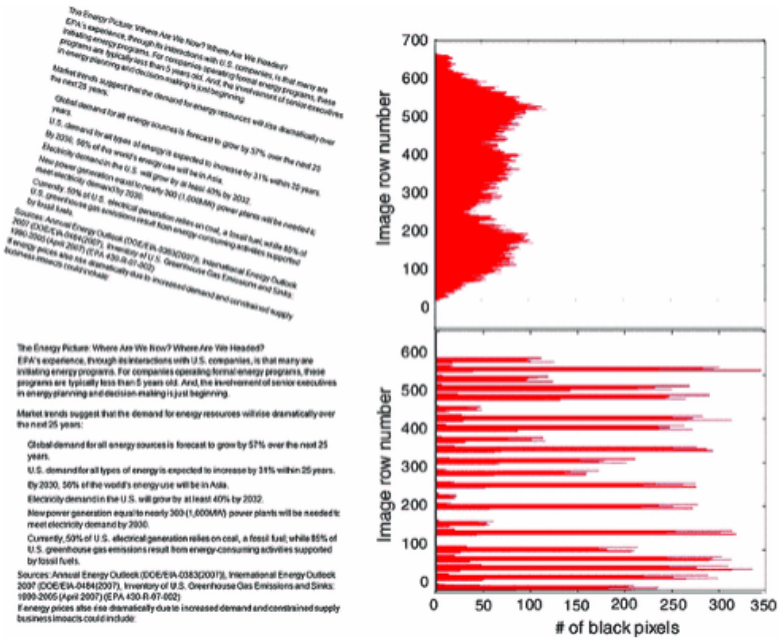


Figure 3.1: Correcting skew using the Projection Profile method.

3.3 Skew correction

Due to the possibility of rotation of the input image and the sensitivity of many document image analysis methods to rotation of the image, document skew should be corrected. Skew detection techniques can be roughly classified into the following groups: Projection profile, Hough transform,Connected components, Clustering, and Correlation between lines techniques.

The method we used is a projection profile method. In this method, the following steps are applied to the binary image :

1. project it horizontally (taking the sum of pixels along rows of the image matrix) to get a histogram of pixels along the height of the image i.e count of foreground pixels for every row.

2. The image is rotated at various angles (at a small interval of angles called Delta) and the difference between the peaks will be calculated (Variance can also be used as one of the metrics). The angle at which the maximum difference between peaks (or Variance) is found, that corresponding angle will be the Skew angle for the image.

3. After finding the Skew angle, we can correct the skewness by rotating the image through an angle equal to the skew angle in the opposite direction of skew.

3.4 Thresholding

Image thresholding is the process of separating the information (objects) of an image from its background, hence, thresholding is usually applied to grey-level or colour document

scanned images. Thresholding can be categorised into two main categories:

- Global thresholding algorithms
- Local or adaptive thresholding

Global thresholding methods choose one threshold value for the entire document image, which is often based on the estimation of the background level from the intensity histogram of the image; hence, it is considered a point processing operation. On the other hand, local adaptive thresholding uses different values for each pixel according to the local area information.
Thresholding is a process of converting a grayscale input image to a bi-level image by using an optimal threshold.
Global thresholding methods are used to automatically reduce a grey-level image to a binary image. The images applied to such methods are assumed to have two classes of pixels (foreground and background). The purpose of a global thresholding method is to automatically specify a threshold value, T, where the pixel values below it are considered foreground and the values above are background.

3.4.1 Otsu's Method

Otsu's Method is a nonparametric and unsupervised method of automatic threshold selection for picture segmentation . An optimal threshold is selected by the discriminant criterion, namely, so as to maximize the separability of the resultant classes in gray levels. The procedure is very simple, utilizing only the zeroth- and the first-order cumulative moments of the gray-level histogram. It is straightforward to extend the method to multithreshold problems.[**11**]
It is indeed one of the widely used techniques used to convert a grey-level image into a binary image then calculates the optimum threshold separating those two classes so that their combined spread (intra-class variance) is minimal: The Otsu method searches for the threshold that minimises the intra-class variance.

3.5 Image preprocessing quality : Structural Similarity Index

In order to know if the pre-processing was efficient , we have to evaluate its performance. For this purpose we are using the Structural Similary Index Measure.
The Structural Similarity Index (SSIM) metric extracts 3 key features from an image:

- Luminance
- Contrast
- Structure

The evaluation of the quality of the image depends on these 3 key features which is why they are used to compute the SSIM measure. The comparison between the two images is performed on the basis of these 3 features.

3.6 Table detection

While extracting text from a document, the data can be present in several ways. And sometimes data under specific structure is not relevant for the extraction process. Similarly for our project to extract data from geological report the table data is not as relevant as the paragraph data. We purely needed to fetch text from paragraphs and ignore the rest of text in different formats. Detecting table is not a straight forward condition while reading an image. In order to achieve that we are trying to implement the following:

- Detecting horizontal and vertical lines in the image.
- Detecting the cells created.

We are performing Morphological transformations to detect the horizontal and vertical lines in an page because it gives us the flexibility to focus on a certain shape and refine those shapes. In our case we wanted to find out the horizontal and vertical lines, combine them and leverage the Contours function to find out the bounding box structures and detect the cells.

3.6.1 Morphological transformations

Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is our original image, second one is called structuring element or kernel which decides the nature of operation. Two basic morphological operators are Erosion and Dilation.

1. Erosion: The basic idea of erosion is just like soil erosion only, it erodes away the boundaries of foreground object (Always try to keep foreground in white). So what does it do? The kernel slides through the image (as in 2D convolution). A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded (made to zero).

   So what happens is that, all the pixels near boundary will be discarded depending upon the size of kernel. So the thickness or size of the foreground object decreases or simply white region decreases in the image. It is useful for removing small white noises (as we have seen in color-space chapter), detach two connected objects etc.

2. Dilation: It is just opposite of erosion. Here, a pixel element is '1' if at-least one pixel under the kernel is '1'. So it increases the white region in the image or size of foreground object increases. Normally, in cases like noise removal, erosion is followed by dilation. Because, erosion removes white noises, but it also shrinks our object. So we dilate it. Since noise is gone, they won't come back, but our object area increases. It is also useful in joining broken parts of an object.

3. Structuring Element: It is an OpenCV function, you just pass the shape and size of the kernel, you get the desired elliptical/circular/rectangular kernel.

3.7 Block text detection

To detect the cells created from the detected horizontal and vertical lines we introduce the concept of contours.

- Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition.

The methods we used to detect cells from an image are as follows:

1. We will first get the entire image dimensions and then using the OpenCV structural element function we will get the horizontal lines.

2. Now, using the erode and dilate function we will apply it to our image and detect and extract the horizontal lines. In the same way, we will repeat these steps to detect the vertical lines by building another kernel.

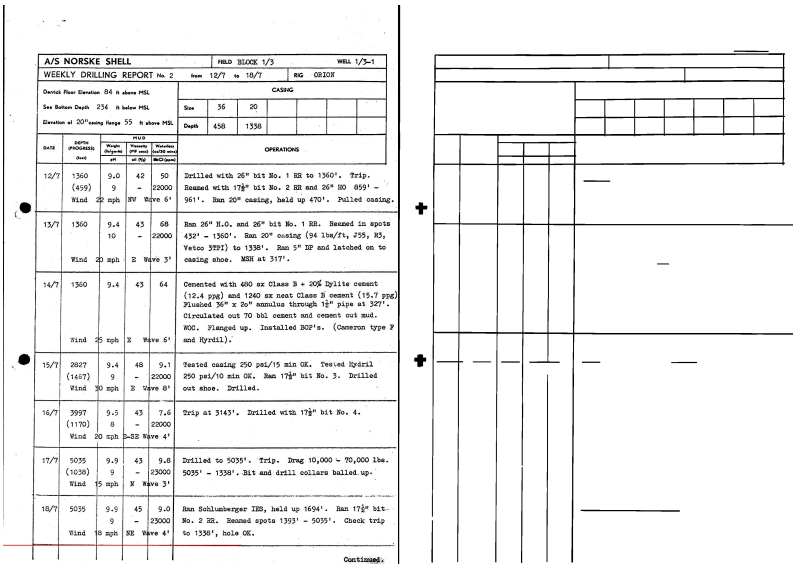
3. Once we have these ready, all we need to do is combine them and make them into a grid-like structure and get a clear tabular representation without the content inside. To do this, we will first combine the vertical and horizontal lines and create a final rectangular kernel. Then we will convert the image back from grey-scale to white.

4. Now that we have our empty table ready, we use Contours to highlight the cell lines and determine the bounding boxes.

5. Next, we will retrieve the dimensions of each contour and store them. Since these contours are rectangles, we will have 4 sides and the position of each cell of the image stored in the list.

6. We count the number of cells detected per page and filter out the images containing cell count more than 25. We have kept the cell count 25 after analysing several documents. We kept a static value because sometimes the contours would have a bounding box around thick underlines and punch holes (punch holes are shown in the scanned copy of the image) and the cell count would increase. We noticed that if cell count is more 25, then the page actually has table content inside it.

Below are examples of our table detection process:



A/S NORSKE SHELL — FIELD BLOCK 1/3 — WELL 1/3-1

WEEKLY DRILLING REPORT No 2 — from 12/7 to 18/7 — RIG ORION

Derrick Floor Elevation 84 ft above MSL
Sea Bottom Depth 234 ft below MSL
Elevation of 20" casing flange 55 ft above MSL

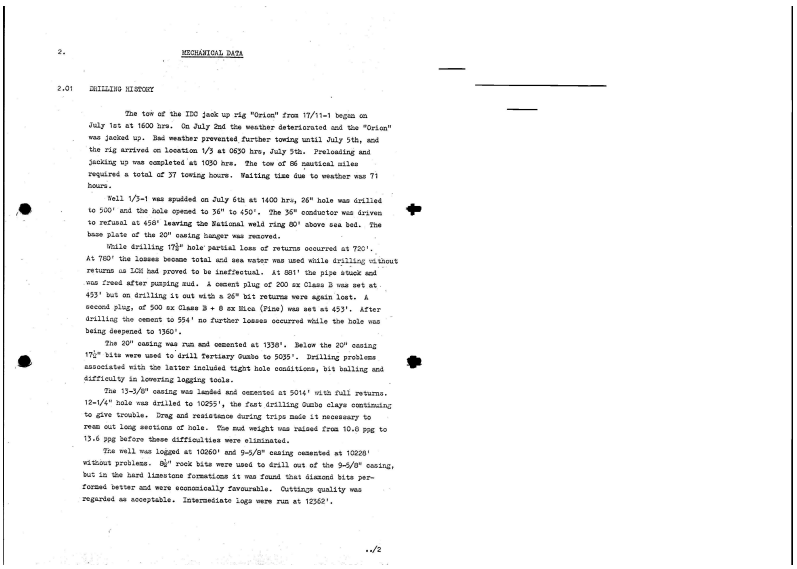
CASING: Size 36 | 20 | | | | | | Depth 458 | 1338

DATE	DEPTH (PROGRESS) (feet)	MUD Weight (lb/gal) at	MUD Viscosity (FV sec) at (fig)	MUD Waterloss (cc/30 min) (API)	OPERATIONS
12/7	1360 (459) Wind 22 mph	9.0 9 -	42 - NW	50 22000 Wave 6'	Drilled with 26" bit No. 1 RR to 1360'. Trip. Reamed with 17½" bit No. 2 RR and 26" HO 859' - 961'. Ran 20" casing, held up 470'. Pulled casing.
13/7	1360 Wind 20 mph	9.4 10 -	43 - E	68 22000 Wave 3'	Ran 26" H.O. and 26" bit No. 1 RR. Reamed in spots 432' - 1360'. Ran 20" casing (94 lbs/ft, J55, BS, Vetco 3IPI) to 1338'. Ran 5" DP and latched on to casing shoe. MSH at 377'.
14/7	1360 Wind 25 mph	9.4 - E	43 - Wave 6'	64	Cemented with 480 sx Class B + 20% Hylite cement (12.4 ppg) and 1240 sx neat Class B cement (15.7 ppg) Flushed 36" x 20" annulus through ½" pipe at 327'. Circulated out 70 bbl cement and cement cut mud. WOC. Flanged up. Installed BOP's. (Cameron type F and Hydril).
15/7	2827 (1467) Wind 30 mph	9.4 9 -	48 - E	9.1 22000 Wave 8'	Tested casing 250 psi/15 min OK. Tested Hydril 250 psi/10 min OK. Ran 17½" bit No. 3. Drilled out shoe. Drilled.
16/7	3997 (1170) Wind 20 mph	9.5 8 -	43 - S-SE	7.6 22000 Wave 4'	Trip at 3143'. Drilled with 17½" bit No. 4.
17/7	5035 (1038) Wind 15 mph	9.9 9 -	43 - N	9.8 25000 Wave 3'	Drilled to 5035'. Trip. Drag 10,000 - 70,000 lbs. 5035' - 1338'. Bit and drill collars balled up.
18/7	5035 Wind 18 mph	9.9 9 -	45 - NE	9.0 25000 Wave 4'	Ran Schlumberger IES, held up 1694'. Ran 17½" bit No. 2 RR. Reamed spots 1393' - 5035'. Check trip to 1338', hole OK.

Continued.

Figure 3.2: Example of cell detection from image containing table.

In the figure [**3.2**] it is noticed that the horizontal and vertical lines are combined and the table structure is extracted.



2. MECHANICAL DATA

2.01 DRILLING HISTORY

The tow of the IDC jack up rig "Orion" from 17/11-1 began on July 1st at 1600 hrs. On July 2nd the weather deteriorated and the "Orion" was jacked up. Bad weather prevented further towing until July 5th, and the rig arrived on location 1/3 at 0630 hrs, July 9th. Preloading and jacking up was completed at 1030 hrs. The tow of 86 nautical miles required a total of 37 towing hours. Waiting time due to weather was 71 hours.

Well 1/3-1 was spudded on July 6th at 1400 hrs, 26" hole was drilled to 500' and the hole opened to 36" to 450'. The 36" conductor was driven to refusal at 458' leaving the National weld ring 80' above sea bed. The base plate of the 20" casing hanger was removed.

While drilling 17½" hole partial loss of returns occurred at 720'. At 780' the losses became total and sea water was used while drilling without returns as LCM had proved to be ineffectual. At 881' the pipe stuck and was freed after pumping mud. A cement plug of 200 sx Class B was set at 453' but on drilling it out with a 26" bit returns were again lost. A second plug, of 500 sx Class B + 8 sx Mica (Fine) was set at 453'. After drilling the cement to 554' no further losses occurred while the hole was being deepened to 1360'.

The 20" casing was run and cemented at 1338'. Below the 20" casing 17½" bits were used to drill Tertiary Gumbo to 5035'. Drilling problems associated with the latter included tight hole conditions, bit balling and difficulty in lowering logging tools.

The 13-3/8" casing was landed and cemented at 5014' with full returns. 12-1/4" hole was drilled to 10255', the fast drilling Gumbo clays continuing to give trouble. Drag and resistance during trips made it necessary to ream out long sections of hole. The mud weight was raised from 10.6 ppg to 13.6 ppg before these difficulties were eliminated.

The well was logged at 10260' and 9-5/8" casing cemented at 10228' without problems. 8½" rock bits were used to drill out of the 9-5/8" casing, but in the hard limestone formations it was found that diamond bits performed better and were economically favourable. Cuttings quality was regarded as acceptable. Intermediate logs were run at 12362'.

../2

Figure 3.3: Example of cell detection from image containing no table

In the figure [**3.3**] it is noticed that the page does not have any tables present in it. So, the extracted image does not contain any table structure.

3.8 Section detection

The idea behind the section detection is to be able to detect the different sections constituting a single page of a document.
The adopted approach to implement it is as follows :

1. Converting image to grayscale and applying a Gaussian Blur
2. Applying an Adaptive threshold
3. Dilate the image to connect adjacent words together
4. Finding contours and drawing bounding box

Chapter 4

Post-processing : Text Analytics

The aim of this part is to analyze the extracted text after the pre-processing steps in order to improve the quality of the output even more but also extract some meaningful information about the extracted textual data. Therefore , this was based on two main steps : First, we implemented a spelling corrector to correct the misspelled words. This is followed by topic analysis which tries to gain some insights from the extracted text using the TF-IDF technique.

4.1 Text correction

Often text extraction is negatively influenced by poor image quality (e.g., scanning resolution, noise) and any mismatch between the instances on which the character image classifier was trained and the rendering of the characters in the printed document (e.g., font, size, spacing). Depending on the language and the image quality of the analyzed image, there will be a different error distribution generated by an OCR process. These errors can be categorized according to the following types:

- Word detection : failing to detect text in the image, commonly caused by poor image quality or text mixed with graphics.
- Word segmentation : failing to bound an individual word correctly, due to wrong interword space detection, generally due to different text alignments and spacing.
- Character segmentation : failing to bound single characters in a segmented word. This is frequent for cursive or connected alphabets.
- Character recognition : failing to identify the correct character for a bounded character image.

Therefore an additional post-processing step is needed to refine the results generated by the extraction.
The algorithm used to achieve this is based on the following sub-steps:

1. Identify the misspelled tokens : as example a misspelled word w
2. Compare the word w with the words in the lexicon
3. Choose the correction c that maximizes the probability p(c|w)
4. Update the text by substituting the chosen correction c to the word

The dictionary lookup compares OCR-output of extraction with the words in a lexicon. The algorithm tries to choose the most likely spelling correction for a given word w . There is no way to know for sure (for example, should "lates" be corrected to "late" or "latest" or "lattes" or ...?), which is why probabilities are used : the idea is to try to find the correction c, out of all possible candidate corrections, that maximizes the probability that c is the intended correction, given the original word w.

However, there are some unseen ("out of vocabulary") words , usually rarely used ones(in our cases the very technical words) that the spelling correction algorithm may fail to correctly predict because the correct word might not appear in the lexicon.

4.2 Topic Analysis

One of the purposes of this project was to gain insightful information from geological reports for better data interpretation. At first, we pre-processed the images so that we can fetch better extracted text with less errors but we wanted to gain some insights regarding the extracted as well. We went forward to see the different text analysis techniques, as using such techniques helps us to extract specific information, like keywords, names, or country information from thousands of words which are written in a document.

Therefore, we chose the technique of word frequency because counting the number of times each word appears in a page is probably the first thing that comes to mind when thinking of text analysis. It is a great statistical measure to calculate because the frequency of a word's use is likely to be a good indicator of its importance. One such technique is the TF-IDF word analysis.

- TF-IDF : Term frequency and inverse document frequency are statistical measures that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents. The metric equation is as follows:

$$tf - idf(t, d, D) = tf(t, d) * idf(t, D)$$

$$tf(t, d) = log(1 + freq(t, d))$$

where *freq(t,d)* is the raw count of a term in a document, i.e., the number of times that term t occurs in document d.

$$idf(t, D) = log(\frac{N}{count(d \in D : t \in d)})$$

N: total number of documents in the corpus $N=|D|N = |D|$ and number of documents where the term t appears.

To achieve our favoured results we go the through the following step:

1. We fetch the extracted text and we are going to preprocess the input text, by removing punctuation and special characters and converting all words to lowercase. The function preprocess() , removes the non-letter characters from the input RDD by using a regular expression.

2. In the next step, we are going to ignore words, such as articles, pronouns, conjunctions. These words have little lexical meaning and express a grammatical relationship with the other words. In many applications, these functions words are not useful and they can be ignored. In text processing applications, function words are part of a list of stop words that include all words that are frequently used and give little semantic contribution to the content of a text. Since there isn't any agreement as to the definition of stop word, it's possible to find many such lists on the Web. And we have used one such list for our data.

3. We transform our RDD to Pair RDDs using the map() function. Pair RDDs are simply RDDs where each element is a key-value pair. They are useful for expressing MapReduce computations.

4. We apply the transformation groupbykey() to the RDD kvwords. The result is a RDD occurrences , where each element is a pair (w,L) , w is a word and L is a list of 1s (as many as the number of occurrences of w). We apply transformation on the new RDD and obtain occurrences in pair (w,len(L)), were len(L) will give the number of occurrences of the respective word.

5. Next, we add the words and their lengths in a dictionary and plot the frequency graph and the wordcloud format for every extracted page from the document. Some plots are shown in the results.

Chapter 5

Experiments and results

5.1 Tools

5.1.1 Tesseract

Tesseract is an optical character recognition engine for various operating systems. It is free software, released under the Apache License. Originally developed by Hewlett-Packard as proprietary software in the 1980s, it was released as open source in 2005 and development has been sponsored by Google since 2006.
In 2006, Tesseract was considered one of the most accurate open-source OCR engines then available.
Tesseract has the ability to recognize more than 100 languages out of the box. It can be trained to recognize other languages. It generally involves the detection of text content on images and translation of the images to encoded text that the computer can easily understand. An image containing text is scanned and analyzed in order to identify the characters in it. Upon identification, the character is converted to machine-encoded text.[**7**]

5.1.2 OpenCV

OpenCV is a library of programming functions mainly aimed at real-time computer vision. OpenCV contains various tools to solve computer vision problems. It contains low level image processing functions and high level algorithms for face detection, feature matching and tracking. On the context of this project it was mainly used for prepocessing the input images in order to improve their quality.

5.1.3 TextBlob

TextBlob is a Python library for processing textual data.It is built on top of NLTK and it provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.[**6**]
This library was used on the spelling correction module to correct spelling mistakes produced by the extraction.

5.1.4 Spark

Apache Spark is a cluster computing framework for parallel data processing that was conceived to address the inefficiencies of Hadoop with respect to iterative computations. Spark is used by both data scientists, who analyze large datasets, and engineers, who

develop data processing applications. Spark allows both to concentrate on their application by hiding all the complexity of running applications in a distributed environment: distributed systems programming, network communication and fault tolerance.

5.2 Results and evaluation

5.2.1 Section detection

The first step after the table detection and image pre-processing is the section detection.

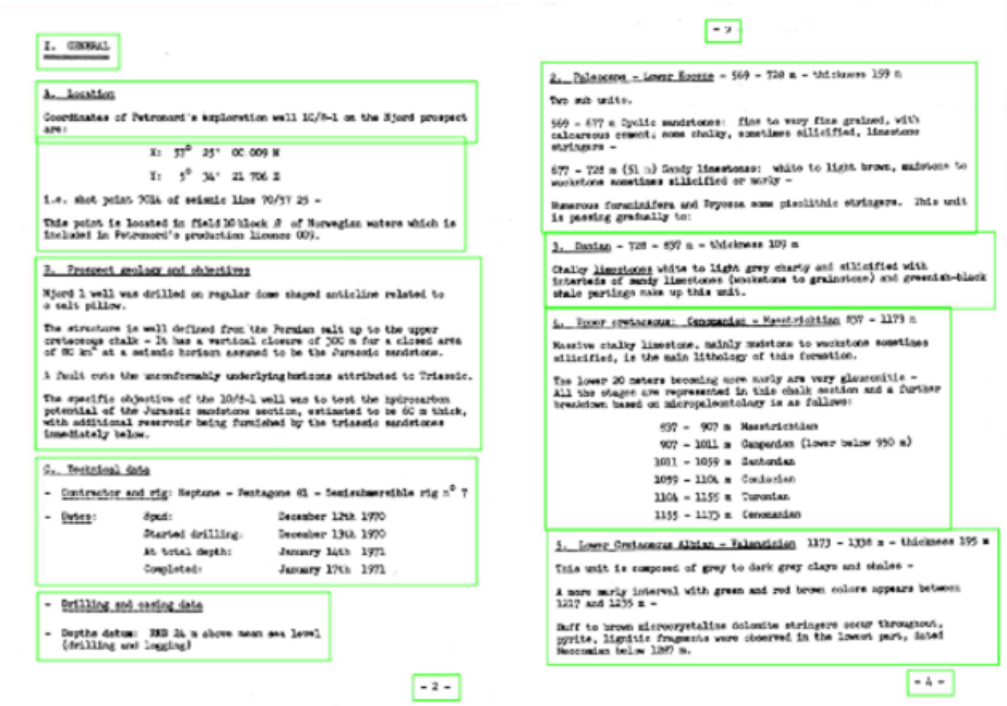


Figure 5.1: Section detection in a page

As the figure [**5.1**] shows , the section are contoured with a green bounding box. Right after this we move forward to extracting the text. But as we have noticed , the output of the extraction still contains a certain number of spelling mistakes and which is why we implemented a text correction module.

5.2.2 Text Correction Evaluation

The figure [**5.2**] above shows an example of the spelling correction in a given page of a document. Many mistakes have been corrected after applying the text correction :

- 'Wopies' : 'Copies'
- 'thohght' : 'thought'
- 'negativg rapults' : 'negative results'

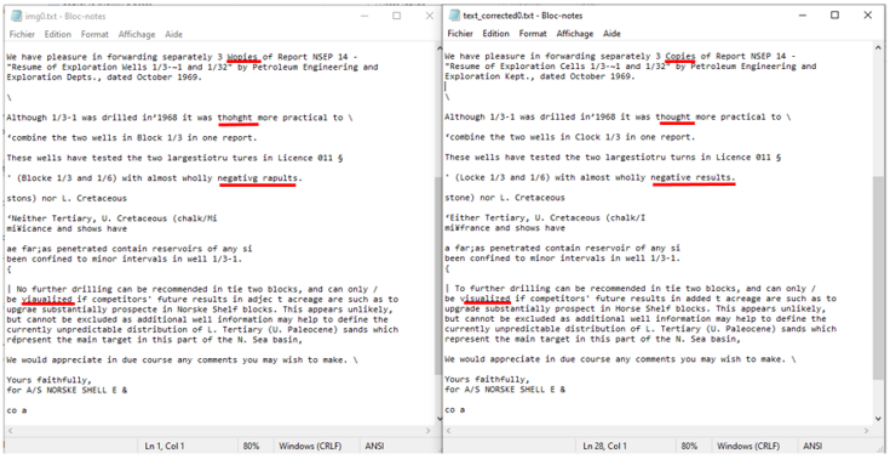


Figure 5.2: Text correction in a page

- 'viaualized' : 'visualized'

Document	Average % of corrected mistakes
Document 1	19,62
Document 2	21,52
Document 3	22,93
Document 4	12,67
Document 5	24,19

Figure 5.3: Text correction evaluation table

The metric that is used to evaluate the text correction model is the average percentage of corrected mistakes avg.

$$avg = \sum_{i=0}^{n-1} \frac{per - i}{n}$$

n : number of pages.
per-i :percentage of corrected mistakes in page i

The table on the figure **[5.3]** shows the results of the average percentage of corrected mistakes metric on a set of documents. The main challenges that we faced for the text correction is the fact that some rare words (technical words related to geology and proper nouns etc.) were present on the texts and could not be corrected after the text correction phase because the correction word does not exist in the reference lexicon.

5.2.3 Word Frequency Evaluation

As we explained earlier, after the text extraction we are going to find the most frequent words for topic analysis and the first thing that we needed to do was to remove the stopwords. The figure [**5.4**] shows an example of the word count difference. On the left hand side the word count is with stopwords and on the right hand side the results shows word count without the stopwords for a page. Words like 'the', 'was','to' etc. are removed from the word list because these words do not add value to our topic analysis.

```
Number of words  350
Top-100 most frequent words
the ---> 23
was ---> 19
to ---> 15
at ---> 15
and ---> 12
of ---> 9
drilling ---> 7
were ---> 6
casing ---> 6
hole ---> 6
```

```
Number of words after stopword removal  181
Top-100 most frequent words
drilling ,  7
casing ,  6
hole ,  6
july ,  5
hrs ,  4
returns ,  4
cemented ,  3
bits ,  3
```

Figure 5.4: Example of Stopwords removal

Below we show our result for the most frequent words according to the TF-IDF technique for per page in a document. We plot a graph for the top 15 words per page. Figure [**5.5**] and [**5.6**] are two examples of our output.

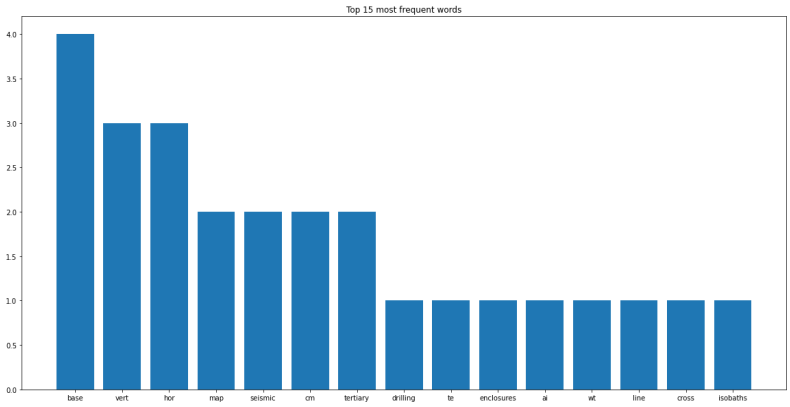


Figure 5.5: Graph 1 of most frequent words in document

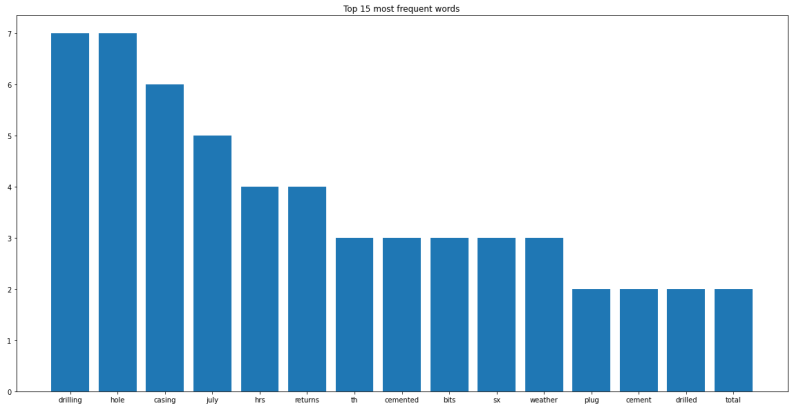


Figure 5.6: Graph 2 of most frequent words in document

Apart from the graph plot we wanted to visualize the words to get an idea about the data. WordCloud is a python library using which we are able to visualize the word analysis. It is a good way to provide a gist about the textual data present in the page. It maps the words on the basis of their occurrences and emphasizes the one's which have occurred more than the rest. Figure [**5.7**] and [**5.8**] are two examples of our output.

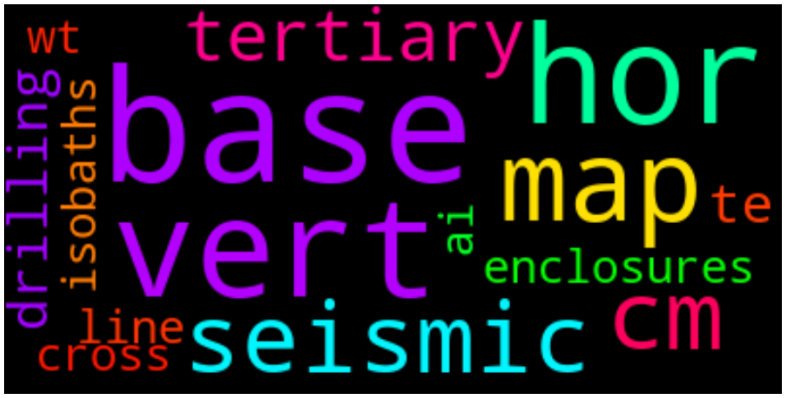


Figure 5.7: Example 1 of most frequent words in document in Wordcloud format

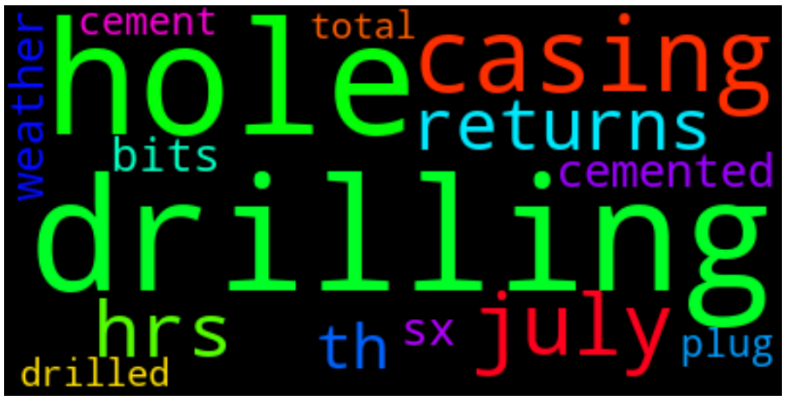


Figure 5.8: Example 2 of most frequent words in document in Wordcloud format

Chapter 6

Conclusion

The objective of this project was to retrieve text efficiently from PDF files and extraction insightful information about the text data.
We began the work by applying numerous image pre-processing steps to improve quality of the images including the table detection that aimed to delete images containing tables as it was requested.Our work for table detection introduced us to morphological transformations and the concept of bounding box. We implemented these techniques for cell detection and to count the number of cells per page to filter images on the basis of it.Going forward we implemented TF-IDF technique for topic analysis and found information regarding the text. But there are more better information retrieval techniques such as Word2Vec, which uses word vectorization to fetch linking information from the text.
One of the main challenges that we faced were the poor quality of certain documents due to different reason : technique of the scan ,different sizes,style,orientation , complex background of image.
A major issue faced during the text correction module was the presence of rare words unknown by the reference dictionary of the used package , those words were either technical words related to the geological field or proper nouns.
Our approach opened up a lot of possibilities for extracting old and tampered PDF documents, which can be pursued forward with upcoming technologies in the field of NLP.

Bibliography

[1] The Github link to our code: *https://github.com/MariaTerrah/BDRP$_{P}$roject*(*Notcited.*)

[2] PAN, Y.-F., HOU, X., LIU, C.-L. *A Robust System to Detect and Localize Texts in Natural Scene Images. The Eighth IAPR International Workshop on Document Analysis Systems* . 2008 (Not cited.)

[3] Jian Liang; DeMenthon, D.; Doermann, D.; *Geometric Rectification of Camera-Captured Document Images*. April 2008. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.30, no.4, pp.591-605. (Not cited.)

[4] Atena Farahmand, Abdolhossein Sarrafzadeh, and Jamshid Shanbehzadeh. *Document Image Noises and Removal Methods* . Proceedings of the International MultiConference of Engineers and Computer Scientists 2013 Vol I, IMECS 2013, March 13 - 15, 2013, Hong Kong (Not cited.)

[5] H. K. Anasuya Devi . *Thresholding: A Pixel-Level Image Processing Methodology Preprocessing Technique for an OCR System for the Brahmi Script* . (Not cited.)

[6] G. Deng and L. W. Cahill. *An Adaptive Gaussian Filter For Noise Reduction and Edge Detection*. Department of Electronic Engineering, La Trobe University Bundoora Victoria 3083 Australia (Not cited.)

[7] Lead developer is Ray Smith, Tesseract OCR github,
`https://github.com/tesseract-ocr/tesseract` (Not cited.)

[8] Image Processing using OpenCV,
`https://docs.opencv.org/3.4/db/df6/tutorial`$_{e}$*rosion*$_{d}$*ilatation.html*(*Notcited.*)

[9] TextBlob : Simplified Text Processing
`https://textblob.readthedocs.io/en/dev/` (Not cited.)

[10] Image Pre-Processing Tool, Olga Miljkovi´c
`http://elib.mi.sanu.ac.rs/files/journals/kjm/32/kjom3209.pdf` (Not cited.)

[11] NOBUYUKI OTSU
`A Threshold Selection Method from Gray-Level Histograms` (Not cited.)

[12] Deselaers, T.; Gass, T.; Heigold, G.; Ney, H.;
`Log-Linear Models for Handwritten Digit Classification. June 2012. IEEE Transactions on Pattern Analysis and Machine Intelligence, , vol.34, no.6, pp.1105-1117, doi: 10.1109/TPAMI.2011.218.` (Not cited.)

[13] Feature Extraction and Transformation - RDD-based API
`https://spark.apache.org/docs/latest/mllib-feature-extraction.html` (Not cited.)

[14] TableNet: Deep Learning model for end-to-end Table detection and Tabular data extraction from Scanned Document Images
`Shubham Paliwal, Vishwanath D, Rohit Rahul, Monika Sharma, Lovekesh Vig`
`TCS Research, New Delhi` (Not cited.)

List of Figures