Your Name: MAYANK SINGH Register No.: 312217205047

DARK DATA EXTRACTION AND ANALYSIS FOR DOCUMENTS

Date: 9 May 2020

Signature(s):

1. Mayank Singh

Mentor Signature

DR. S. KARTHIKA

DR. N. BHALAJI

ABSTRACT

Dark data is the data which is acquired by organizations in the normal course of their operations but not used in any manner to derive insights or for decision making and further business logics. However, the same data could be useful to plan product roadmaps, aid business decisions or optimize operations. This is possible today due to modern techniques of machine learning and data analytics.

The term "dark" does not refer to something evil or illegal. Nor is it specifically about security or privacy. Rather, it's about data that's hidden from view, easy to ignore, and hard to access or analyze. It can include information gathered by sensors, telemetic, feedbacks, surveys, financial information, logs, inactive and Old Versions of Relevant documents, documents and great mass of data buried in images, text, tables, figures, PDFs, csv, mp3, ogg, xlsx, png, doc etc.

Our aim is to **extract** dark data and convert images having text, non-searchable and scanned PDFs into searchable PDFs and arranged them according to the relevance to a given search query. And information contained in these documents can be further useful for textual analysis and visualization.

1. INTRODUCTION

Dark data is the data which is acquired by organizations in the normal course of their operations but not used in any manner to derive insights or for decision making and further business logics. The term "dark" does not refer to something evil or illegal. Nor is it specifically about security or privacy. Rather, it's about data that's hidden from view, easy to ignore, and hard to access or analyze. Dark data is analogous to Dark Matter in Physics.
It can include information gathered by sensors, telemetic, feedbacks, surveys, financial information, logs, inactive and Old Versions of Relevant documents etc. Great mass of data buried in images, text, tables, figures, PDFs, csv, mp3, ogg, xlsx, png, doc etc. There is still ongoing research in the field of Dark Data.

This project is intended to convert images having text, non-searchable and scanned PDFs into searchable PDFs and arranged them according to the relevance to a given search query. And information contained in these documents can be further useful for textual analysis and visualization.

Like in our college (SSN College of Engineering) itself so much of data is stored in form of dark data which is not in use at all. That data is simply taking up the space and energy.
This data includes Old question papers, pass out student’s records, faculty details, old notes and several information related to non-teaching staff which are currently of not any use and thus can be called as Dark Data. But if we are able to process this data and derive some insight like analyzing old question papers can predict that what could be the pattern and type of questions in upcoming exams. This can lead to better performances in academics. Similarly, in any other fields.

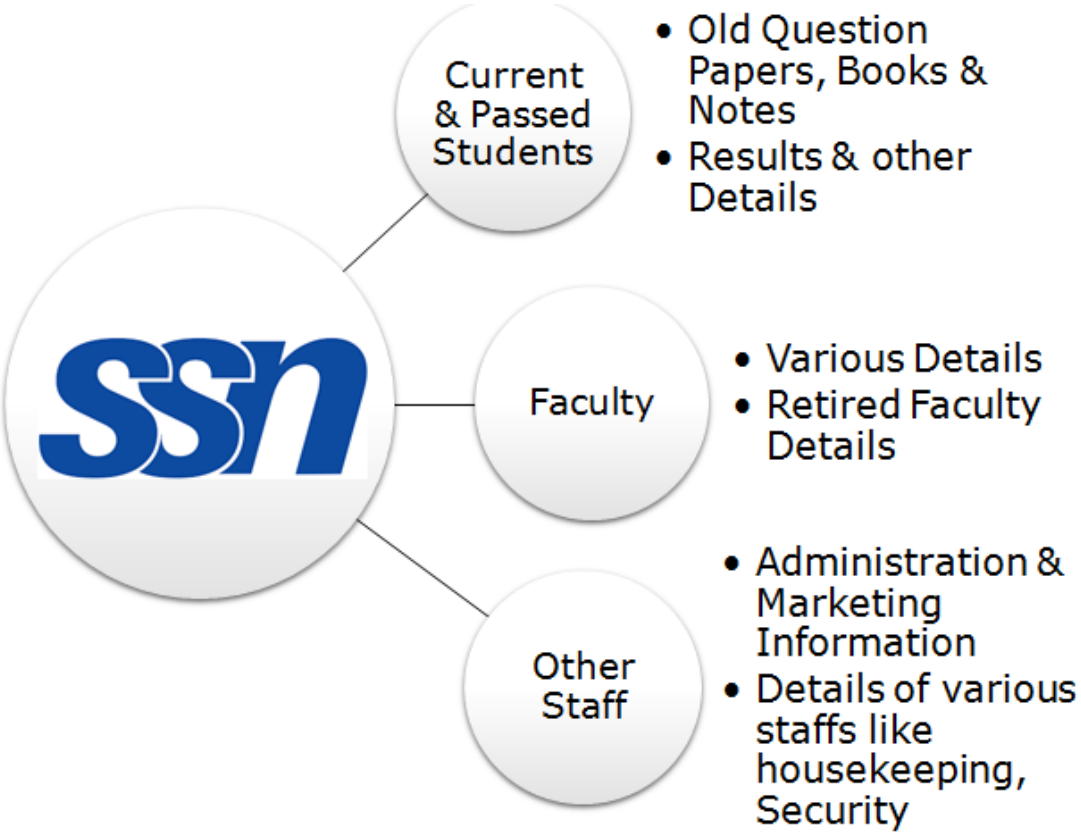


FIGURE 1.1: Dark Data Example

PROBLEM STATEMENT:

A lot of data acquired by organizations is stored in form of scanned images, non-searchable PDFs and scanned images having PDFs, etc. So, our aim is to convert these documents into searchable PDFs and rank and arrange them according to their relevance to a search query given by the user.

The problem is approached in following ways:

1. Scanned Images to editable text file using ***pytesseract*** - A wrapper for Google's Tesseract-OCR Engine.
2. Text Extraction from PDFs having scanned images and non-searchable text to list (collection in Python) of Keywords/text using Amazon's ***textract***.
3. Ranking matching documents according to their relevance to a given *search query* using ***Okapi BM25.***

2. LITERATURE SURVEY

SURVEY RELATED TO Tesseract-OCR:

- Ray Smith. Google Inc.," An Overview of the Tesseract OCR Engine"

Tesseract never needed its own page layout analysis. Tesseract therefore assumes that its input is a binary image with optional polygonal text regions defined. Processing follows a traditional step-by-step pipeline, but some of the stages were unusual in their day, and possibly remain so even now. The first step is a connected component analysis in which outlines of the components are stored. This was a computationally expensive design decision at the time, but had a significant advantage: by inspection of the nesting of outlines, and the number of child and grandchild outlines, it is simple to detect inverse text and recognize it as easily as black-on-white text. Tesseract was probably the first OCR engine able to handle white-on-black text so trivially. At this stage, outlines are gathered together, purely by nesting, into Blobs. Blobs are organized into text lines, and the lines and regions are analyzed for fixed pitch or proportional text. Text lines are broken into words differently according to the kind of character spacing. Fixed pitch text is chopped immediately by character cells. Proportional text is broken into words using definite spaces and fuzzy spaces. Recognition then proceeds as a two-pass process. In the first pass, an attempt is made to recognize each word in turn. Each word that is satisfactory is passed to an adaptive classifier as training data. The adaptive classifier then gets a chance to more accurately recognize text lower down the page. Since the adaptive classifier may have learned something useful too late to make a contribution near the top of the page, a second pass is run over the page, in which words that were not recognized well enough are recognized again. A final phase resolves fuzzy spaces, and checks alternative hypotheses for the x-height to locate small cap text.

SURVEY RELATED TO textract:

- Dean Malmgren, 14 Nov 2019, "extract text from any document. no muss. no fuss."

textract isn't the first project with the aim to provide a simple interface for extracting text from any document. But this is, to the best of my knowledge, the only project that is written in python (a language commonly chosen by the natural language processing community) and is method agnostic about how content is extracted I'm sure that there are other similar projects out there, but here is a small sample of similar projects:

- Apache Tika has very similar, if not identical, aims as textract and has impressive coverage of a wide range of file formats. It is written in java.
- textract (node.js) has similar aims as this textract package (including an identical name! great minds...). It is written in node.js.
- pandoc is intended to be a document conversion tool (a much more difficult task!), but it does have the ability to convert to plain text. It is written in Haskell.

SURVEY RELATED TO BM25:

- Billel Aklouche, brahim Bounhas, Yahya Slimani, 03 October 2019, "BM25 Beyond Query-Document Similarity."

The massive growth of information produced and shared online has made retrieving relevant documents a difficult task. Query Expansion (QE) based on term co-occurrence statistics has been widely applied in an attempt to improve retrieval effectiveness. However, selecting good expansion terms using co-occurrence graphs is challenging. In this paper, we present an adapted version of the BM25 model, which allows measuring the similarity between terms. First, a context window-based approach is applied over the entire corpus in order to construct the term co-occurrence graph. Afterward, using the proposed adapted version of BM25, candidate expansion terms are selected according to their similarity with the whole query. This measure stands out by its ability to evaluate the discriminative power of terms and select semantically related terms to the query. Experiments on two ad-hoc TREC collections (the standard Robust04 collection and the new TREC Washington Post collection) show that our proposal outperforms the baselines over three state-of-the-art IR models and leads to significant improvements in retrieval effectiveness.

3. MOTIVATION

- A study by VERITAS involving 22 countries shows that 52% of all data collected by organizations is dark. They say it is data "**beneath the line of sight of senior management**".
- IBM reports that 90% of IoT sensor data is not used.
- Moreover, 60% of this data loses its value within milliseconds.
- Technologies that allow us to process this data in almost real time is therefore critical. If not, we end up with dark data.
- According to the New York Times, 90% of energy used by data centers is wasted in storing this data.
- There is no such full-fledged project that converts non-searchable documents into searchable documents and arranges or ranks them according to given query search.

4. PROPOSED METHODOLOGY

We have divided the problem into multiple sub-problems namely,

1. Images to Text,
2. Non-searchable and PDFs having images to searchable PDFs,
3. Ranking documents according to *search query*.

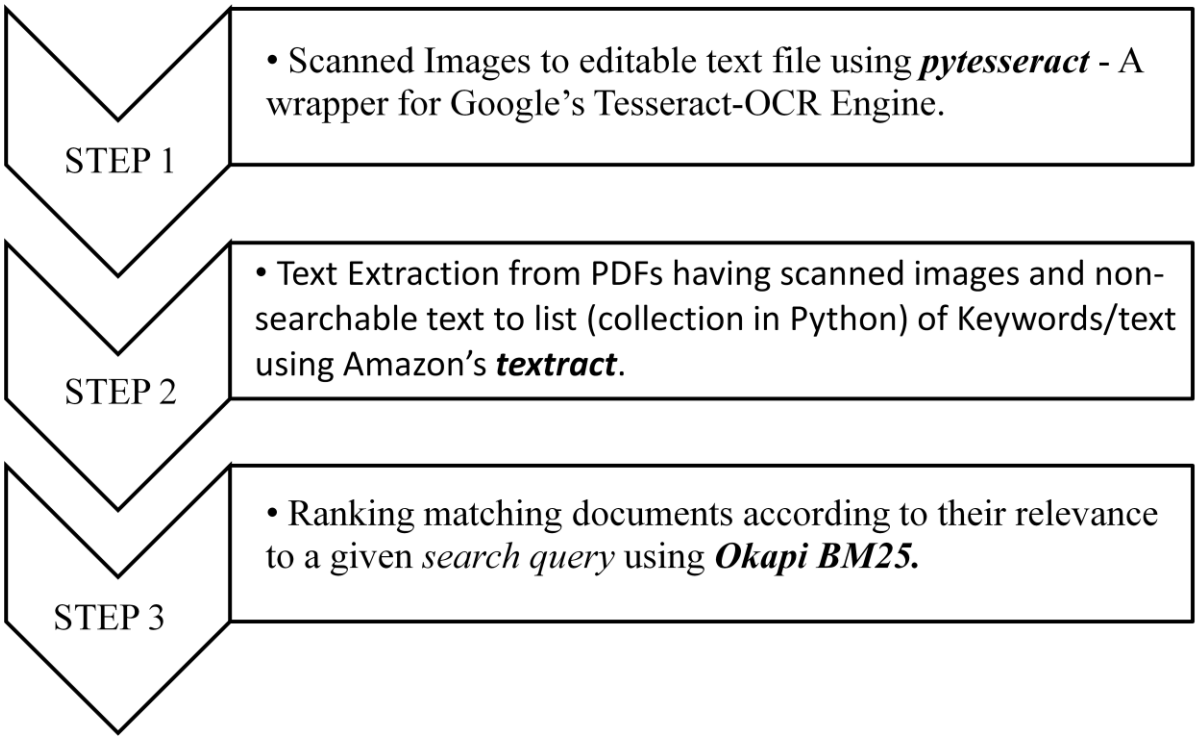


FIGURE 4.1: Approach to the problem

STEP 1: Image to Text (using pytesseract)

Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and "read" the text embedded in images.

Python-tesseract is a wrapper for Google's Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file.

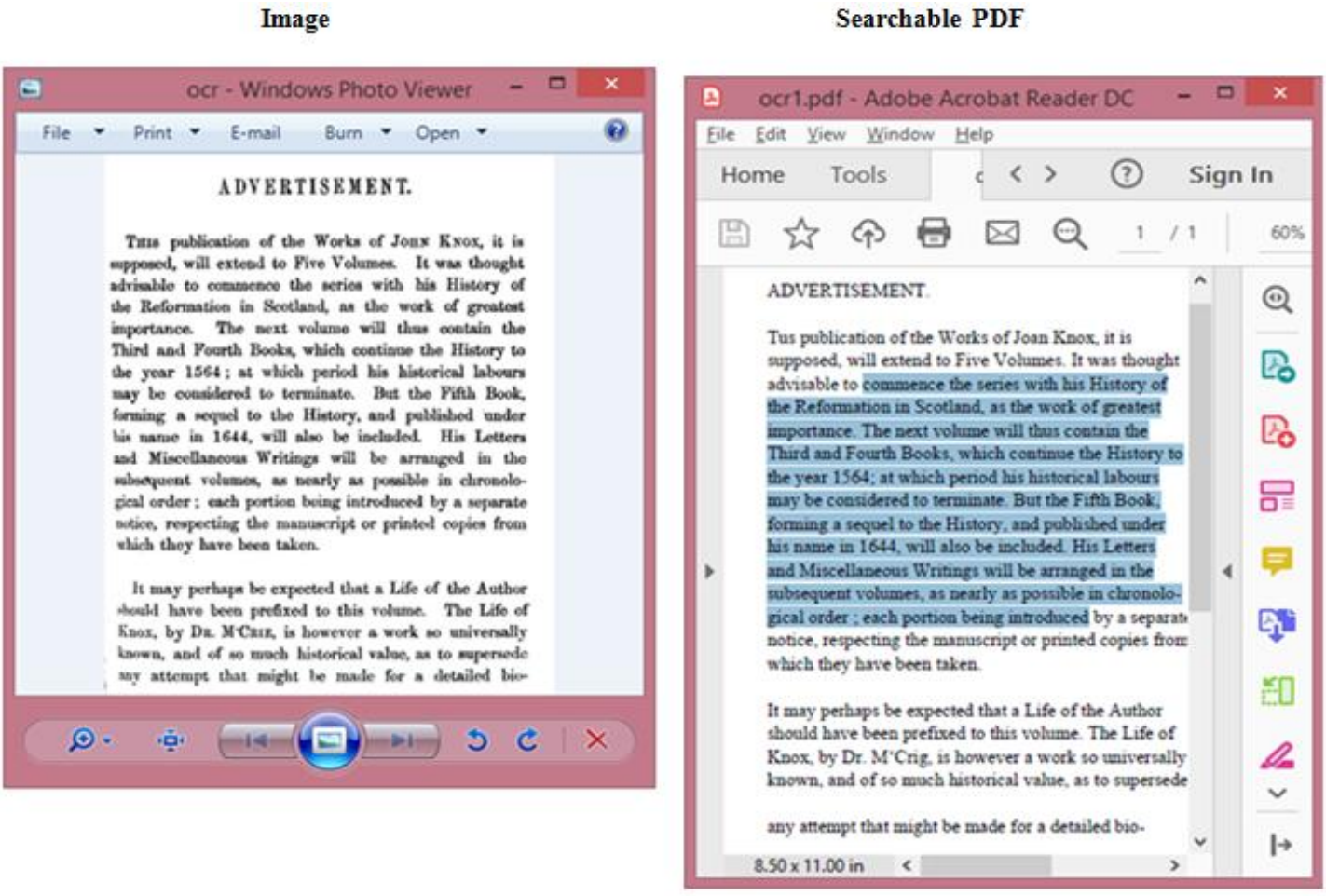
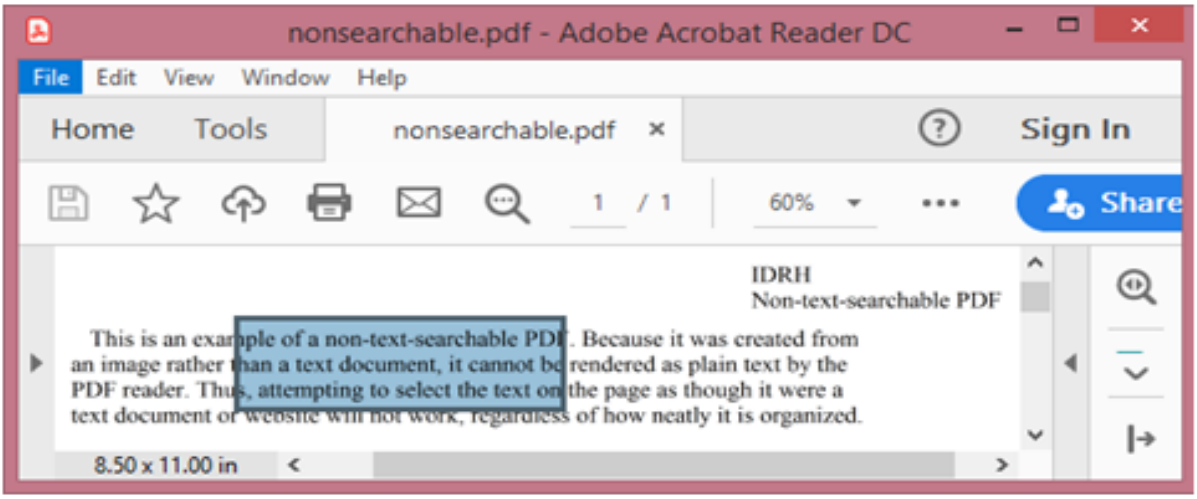


FIGURE 4.2: Converted Image to Searchable PDF

STEP 2: Non-searchable and PDFs having images to searchable PDFs (using textract)

As undesirable as it might be, more often than not there is extremely useful information embedded in Word documents, PowerPoint presentations, PDFs, etc—so-called "dark data"—that would be valuable for further textual analysis and visualization. While several packages exist for extracting content from each of these formats on their own, this package provides a single interface for extracting content from any type of file, without any irrelevant mark-up.

Non-Searchable PDF:



Searchable PDF:

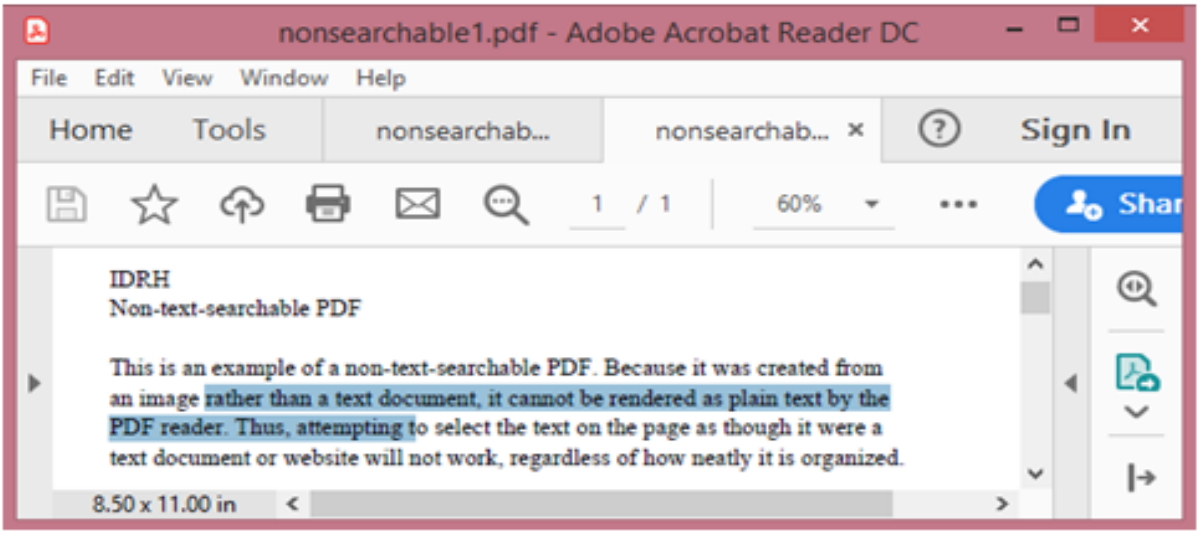


FIGURE 4.3: Converted Non-Searchable PDF to Searchable PDF

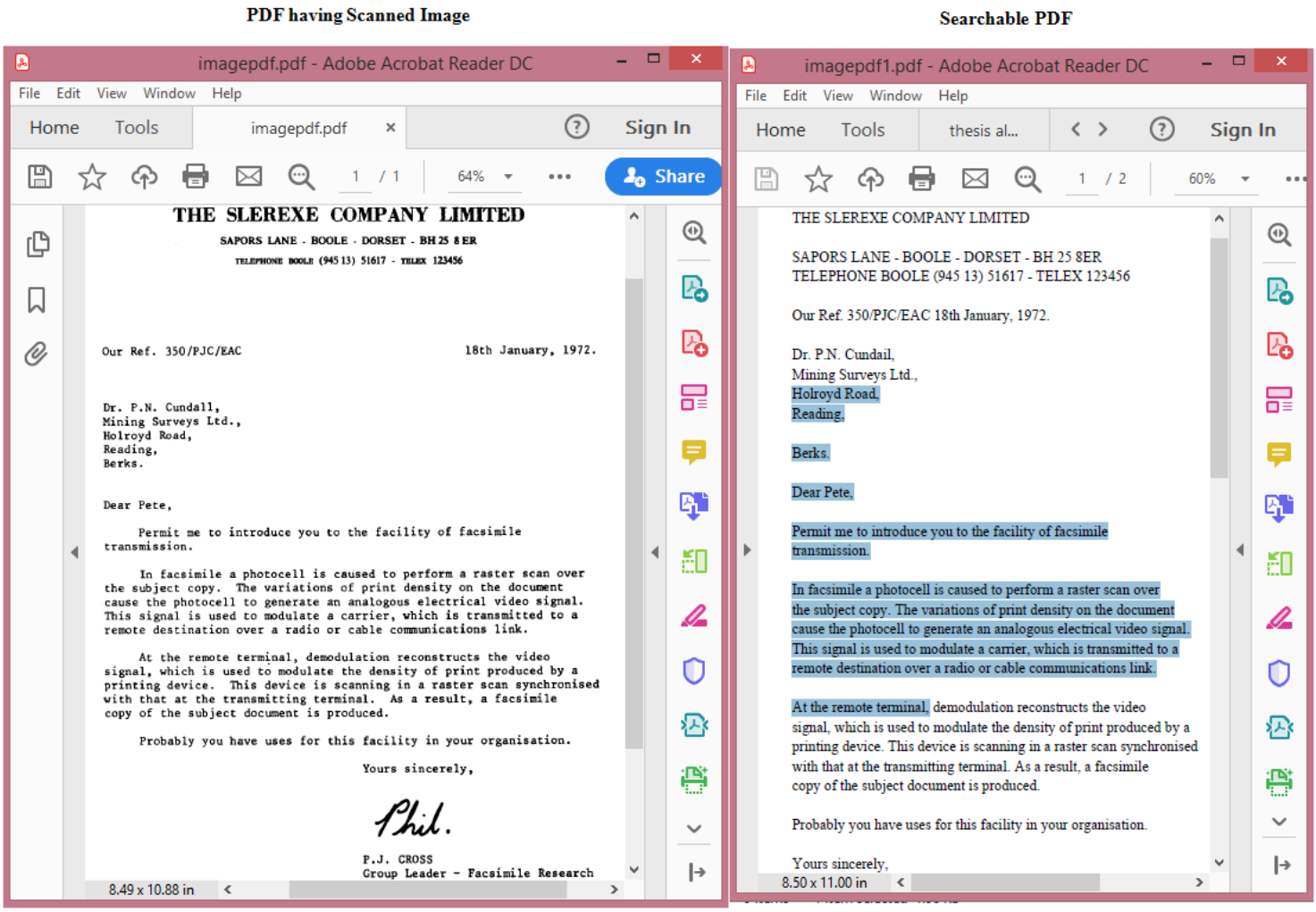


FIGURE 4.4: Converted PDF having scanned Images to Searchable PDF

STEP 3: Arranging Documents according to search query (using okapi BM25 function)

1. The Traditional Probabilistic Weighting Scheme:

In its most general form, the traditional probabilistic term weighting function is:

$$\frac{((k_3+1)\times q)}{(k_3+q)} \times \frac{((k_1+1)\times f)}{(k_1\times L+f)} \times \frac{\log\left((r+0.5)\times(N-n-R+r+0.5)\right)}{((n-r+0.5)\times(R-r+0.5)} \quad \text{.....Eqn(1)}$$

Where,

- k_1, k_3 are constants,
- q is the wqf, the within query frequency,

- f is the wdf, the within document frequency,
- n is the number of documents in the collection indexed by this term,
- N is the total number of documents in the collection,
- r is the number of relevant documents indexed by this term,
- R is the total number of relevant documents,
- L is the normalised document length (i.e. the length of this document divided by the average length of documents in the collection).

The factors (k_3 + 1) and (k_1 + 1) are unnecessary here, but help scale the weights, so the first component is 1 when q = 1 etc. But they are critical below when we add an extra item to the sum of term weights.

2. **BM11**

BM11 uses formula Eqn(1) for the term weights, but adds an extra item to the sum of term weights to give the overall document score:

$$\frac{k_2 \times n_q \times (1-L)}{(1+L)} \text{.....} \textbf{Eqn(2)}$$

where:

- n_q is the number of terms in the query (the query length),
- k_2 is yet another constant.

Note that this extra item is zero when L = 1.

3. **BM15**

BM15 is BM11 with the $k_1 + f$ in place of $k_1 L + f$ in Eqn(1).

4. **BM25**

BM25 combines the B11 and B15 with a scaling factor, b, which turns BM15 into BM11 as it moves from 0 to 1:

$$\frac{((k_3+1)\times q)}{(k_3+q)} \times \frac{((k_1+1)\times f)}{(K+f)} \times \frac{\log((r+0.5)\times(N-n-R+r+0.5))}{((n-r+0.5)\times(R-r+0.5)} \text{.....Eqn(3)}$$

where:

$K = k_1 \times (b \times L + (1-b))$

BM25 originally introduced another constant, as a power to which f and K are raised. However, the powers other than 1 were *'not helpful'*, and other tests confirm this, so Xapian's implementation of BM25 ignores this.

Eqn(2) and Eqn(3) make up BM25.

This does all seem somewhat ad-hoc, with so many unknown constants in the formula. But note that with $k_2 = 0$ and $b = 1$ we get the traditional formula anyway.

The default parameter values Xapian uses are $k_1 = 1$, $k_2 = 0$, $k_3 = 1$, and $b = 0.5$. These are reasonable defaults, but the optimum values will vary with both the documents being searched and the type of queries, so you may be able to improve the effectiveness of your search system by tuning the values of these parameters.

In Xapian, we also apply a floor to L (0.5 by default) which helps stop tiny documents get ridiculously high weights. And the matcher wants the extra item in the sum to be positive, so we add $k_2 n_q$ (constant for a given query) to Eqn(2) to give:

$$\frac{(2\times k_2 \times n_q)}{(1+L)} \text{ .....Eqn(4)}$$

Example explanation:

The given FIGURE 3.5 shows the output of bm25 implementation:

No. of Documents Used: 3

Search Query: "text"

Input Documents Name	Count (Occurrences of “text”)
1. MiniProject_abstract	1
2. testings	6
3. textpdf	40

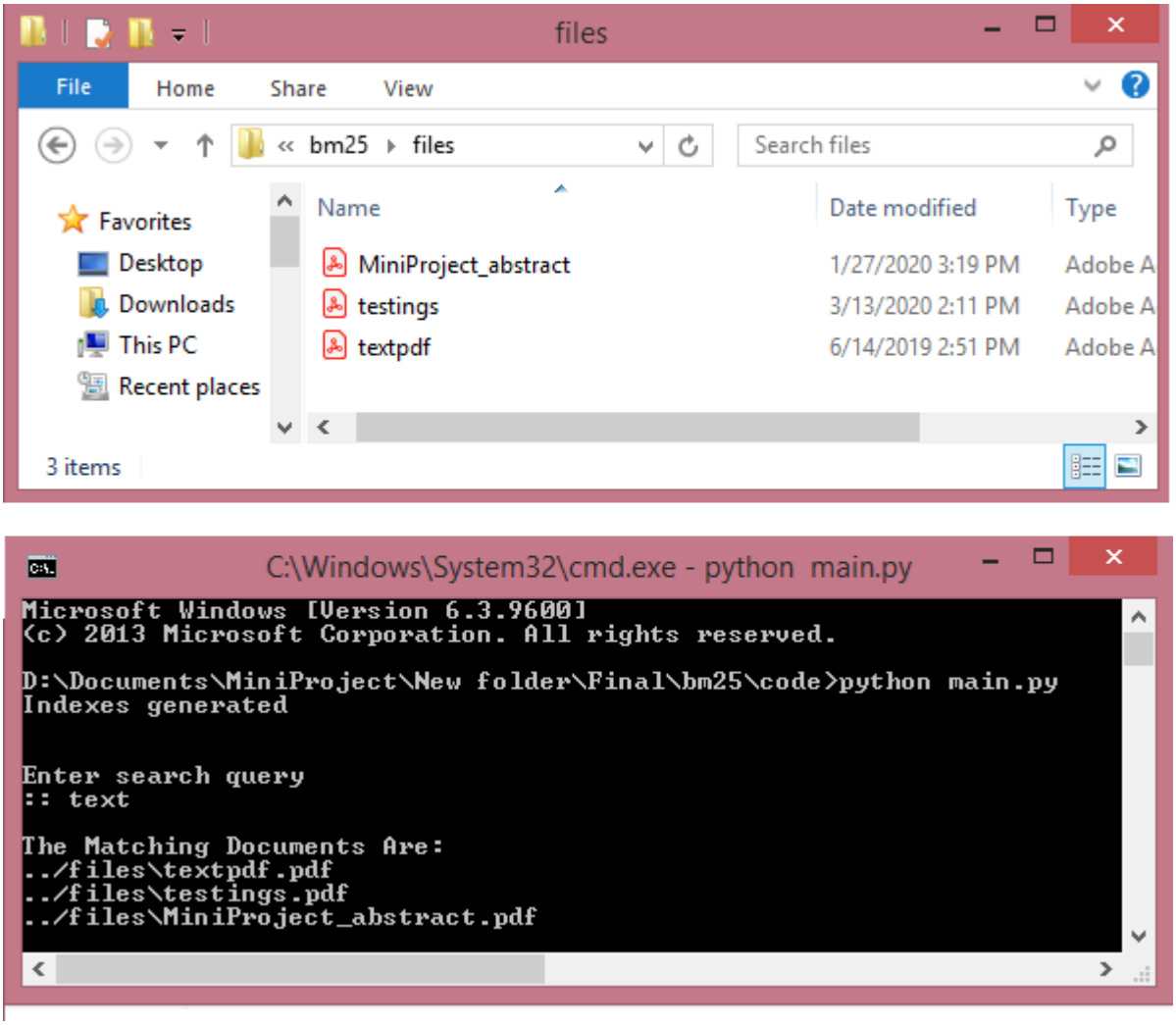


FIGURE 4.5: Ranking documents according to search query

Hence, the document has been ranked and arranged according to the occurrence of word “text”.

5. EXPERIMENTAL RESULTS AND DISCUSSION

1. Project uses two OCR to convert images and documents into searchable PDFs:

   Google's Tesseract (Accuracy around 96.09%)
   Amazon's Textract (Accuracy around 97%)

   OUTCOME: Project's OCR Accuracy: Around 96.9%

2. Two techniques were evaluated BM25 and TF-IDF to weight the terms on documents. The experiments show that TF-IDF improves the performance evaluation of feature extraction according to the maximum value of F1-measure is 89.77 for TF-IDF and 89.16 for BM25.

6. CONCLUSION

In this documentation, we have discussed in detail the conversion of images having text using pytesseract - A wrapper for Google's Tesseract-OCR Engine, non-searchable and scanned PDFs into searchable PDFs to list (collection in Python) of Keywords/text using Amazon's textract and arranged them according to the relevance to a given search query using the best match ranking algorithm Okapi BM25**.**

We have learnt and implemented the BM25 algorithm along with comparison between BM25 and TF-ID model and its advantages.

7. FUTURE ENHANCEMENT

Currently, this project extract data from text documents only but it can be further extended for different types of documents such as audio and video files.

Analysis of this data can be done using different algorithms. Deep Leaning Models can be used for very large data sets instead of classical machine learning model for document categorization. Deduce the performance of different leaning models. Data which cannot be further structured, meta data labels can be attached. Technologies that allow real-time possessing of data can be researched upon and created.

8. REFERENCES

- Bisson, Simon, May 31 2019, "Using machine learning to solve your dark data nightmare."
- Dayley, Alan, March 28 2013, "Innovation Insight: File Analysis Innovation Delivers an Understanding of Unstructured Dark Data."
- Team Data Tree May 02, 2019. "What Is Dark Data?"
- Datumize. 2019. "The Evolution of Dark Data and how you can harness it to make your business Smarter."
- Dayley, Alan. 2013. "Innovation Insight: File Analysis Innovation Delivers an Understanding of Unstructured Dark Data."
- Sarang Saxena, St. Joseph's Degree & Pg College, January 2018, "Dark Data and Its Future Prospects"
- Vinay R. Rao, March 08, 2018 "How data becomes knowledge, Part 3: Extracting dark data"
- Thomas LaMonte, Mar 16, 2018, "What is Dark Data: 7 Questions you were afraid to ask."
- DeepDive, Stanford University. 2017. "DeepDive"
- Devopedia. January 6 2020. "Dark Data." Version 4,
- Accenture, February 19 2019. "What is Dark Data?" AI 101
- E. Robertson and K. Spärck Jones, "Relevance weighting of search terms."
- Thayyaba Khatoon¸ May 2018, "Query Expansion with Enhanced-BM25 Approach for Improving the Search Query Performance on Clustered Biomedical Literature Retrieval."