Assignment 1

Björn Bulkens, Klemens Gerber, Daniel M. Knorr

November 13, 2022

1. Text Analytics Pipeline

   (a) Transition from physical to digitized documents:

      i. Drivers:
      Drivers for a text analytics project in this environment are mainly quality of service and cost related. Large amounts of text data are generated in insurance companies and these amounts may not be analyzed by hand if cost and time are considerable factors. Therefore, as a means of optimizing cost and time usage, an automated analysis could be very useful. Also the unification of the analysis makes results much more usable generally than a painstaking analysis by hand, that might include human errors.

      ii. objectives:
      The objective of the text analysis project should be to extract information from the available data, that is as complete as it has to be. To increase the probability of meeting the customer needs, a data pipeline has to be developed that can easily make a means of analyzing these needs available to the employees of the insurance company. Retaining data quality and timeliness for new data should therefore all be of high priority for a project like this.

      iii. Data availability:
      Data availability could pose a major problem to a project like this. As described, the insurance company recently transitioned from print outs to digital documents. To actually get any usable results, these physical documents have to be digitized in a manner, that makes them usable for our text analytics task. Therefore the data availability is likely a roadblock for the project until digitization has been completed. Live data will not be the problem but historic data could turn out to be sparse.

      iv. cost factors:
      Cost factors can be large hindrances for a text analytics project. If the large amount of physical text data in form of print outs does indeed have to be digitized, this step alone could eat up a large amount of the project budget, as there is no easy way of automating this. Most likely employees will have dedicate significant amounts of time and therefore money to scanning these documents and to storing them centrally in an organized manner. After this the regular hurds of a text analytics project have to be dealt with, meaning computational infrastructure as well cost for software and working hours of the team actually working on the project. These are all significant cost factors that have to be considered.

   (b) Elasticsearch vs. relational DB

      i. Advantages of Elasticsearch:
      Search possible without exact match (search term misspelled) and full text search are possible; Relevance based result sorting included

      ii. Disadvantages of Elasticsearch:
      Classical disadvantages of NoSQL storage vs relational SQL databases: No joins in NoSQL, SQL is better when consistency is critical, vertical scalability of SQL is better

(c) 2 (dis)advantages for each of the following concepts

i. Stop word removal:
+ Decreasing corpus size making the actual analytics more efficient
+ Focus on important word and therefor more concentrated information
- Some tasks need stop words, like sentiment analysis (negations can not be removed)
- Context of sentences is lost

ii. Stemming:
+ It reduces complexity of the vocabulary
+ Words become comparable independent of casus, gender and time
- Stemms can be ambiguous, lemmatization can be a better option then
- Stemming has to be tuned otherwise under or overstemming can lead to problems

2. Regular Expressions

(a) Basic Regex Parser (can be found in the jupyter notebook)

(b) RegEx for HTML:
Regular expressions could be a valuable tool to filter HTML syntax when parsing HTML files for text analytics purposes. DOM elements are predictable structure elements that can be found with RegEx and individual elements can then be further processed. While this is certainly possible, we personally would prefer to use a library specific for working with HTML-formats like e.g. beautiful soup for such a task, as it is a much more specialized tool. After deleting the unnecessary syntax element it would be viable to use Regular Expressions for other data cleansing or data exploration purposes, before conducting further analysis, as they are a very lightweight and powerful tools. But only relaying on RegEx for that would be to much effort for something that can be solved efficiently and fast in another way.

3. Information Extraction from PDF Documents:

(a) We used pdfplumber and PyPDF2 as the two Methods to convert file contents to plain text.

The quantitative analysis includes the run-time measurement. PyPDF2 converts the file contents (for all files) to plain text in much less time than pdfplumber. In the Code the run-time measurements are in the specific .txt-Generation parts (see "Assignment1_P1-3.1.ipynb").

The qualitative analysis includes importing the .pdf-file into Adobe Acrobat Reader and copying the contents from there into a .txt-File. We assume that Adobe works well, so we calculate the similarity between the generated Files (with pdfplumber and PyPDF2) and the

"bahnstadtAdobeAcrobatReaderAllesMarkiert.txt"-File. As an example, we calculate the similarity for the generated bahnstadt-Files. The result is that the generated File with pdfplumber has a similarity of "0.053. . . " but the generated File with PyPDF2 has a similarity of "0.262. . . " which is much higher. In the Code the similarity measurements are in the SequenceMatcher() part (see "Assignment1_P1-3.1.ipynb").

PyPDF2 is in both analysis better than pdfplumber and consequently we use the generated Files by PyPDF2. The only exception is the processing of the tabular bundeswehr.pdf-file. This was specially processed with pdfplumber and Panda. So in Problem 1-3.3(i and ii) we use the bundeswehr_daten.txt-File and not the bundeswehr_PyPDF2.txt-File.

The code of this task is provided in jupyter notebook ("Assignment1_P1-3.1.ipynb").

(b) Why is a quality conversion from pdf to plain text hard?
A quality conversion from PDF to plain text is not trivial to provide. This is due to the variability of information in the PDF format. To convert a pdf to plain text without losing structural information, there has to be a way to account for information, that is given via formal factors like text size or color, as pdfs often rely on information like that to get information across. Also, it is not trivial to analyse a PDFs structure, as the factors that signify a subsection or a sub sub section often vary between pdfs of different sources, and even in pdfs that stem from the same source.

This makes especially the conversion of a large number of pdfs, possibly even from different persons and points in time like it is often the case in everyday settings, very difficult, as the format, even if it can be recognized consistently, often varies greatly between pdfs making the conversion of information from pdf to plain text extremely difficult, time, and resource intensive, or resulting in mostly bad data quality.

(c) The results of this task are provided in the juypter notebook ("Assignment1_P1-3.3.ipynb")

(d) The solution works not well, especially for the double_ocr.pdf-File. In this file the phone numbers appear directly after the house numbers which is difficult to split. In the Problem1-3.4.txt-File are the extracted phone numbers for the two .pdf-Files. An example: In the double_ocr.pdf-File is the phone number "27226". Directly in front of the phone number is the number of the address which is "14". The parsing thinks, that "14" is the beginning of the phone number and because of this "1427226" is inserted into the Output-File instead of the correct phone number "27226". Another example is "1453700" instead of the correct phone number "53700". The parsing works better for the single_ocr.pdf-File, because the phone numbers are separated from the address numbers by many points ("....."). For example in front of the phone number "280281" is the address number "17", but separated by many points. With the solution in P1-3.3(i) these points are identified as a separation that the correct phone number is inserted into the output-File. However, not all phone numbers are recognized from the single_ocr.pdf-File.

The code of this task is provided in jupyter notebook ("Assignment1_P1-3.4.ipynb").