Assignment 1

Felix Hausberger

Christopher Klammt

Nils Krehl

Problem 1-1 Text Analytics Pipeline

1.

(i) The driver for this problem is difficult to identify. It could either be a text-driven or problem-driven analysis.
Since it is a completely new product, one can assume no information about the product is known and therefore no hypothesis can be made in advance, which leads to a normal text-driven approach as one is interested in the pure data. But as we probably assume, that the user feedback contains complaints regarding things they don't like as well as positive feedback, a problem-driven analysis could also make sense. The background theory, that the data contains helpful feedback for further development could represent a hypothesis, but is therefore maybe a bit too generic as we don't try to prove concrete feedback aspects. So in this case the text summarization or topic detection tasks could rather be classified as a normal text-driven analysis.

(ii) This problem clearly is method-driven, as we found out about a new method and want to implement it into our current pipeline.

2.

Advantages:

- One can use different sampling rates for the different subgroups if a certain subgroup is deemed as more important (for example a certain age or income group).
- On the other hand in case the complete data set is unbalanced due to certain subgroups one can easily create balanced samples
  with stratified sampling
- One can as well receive results for each subgroup separately if needed/wanted
- It should generally reduce the sampling error

Disadvantages:

- It is more time- and resource-consuming, because of the extra overhead of forming the subgroups
- It is not straight-forward to apply in case instances cannot be assigned to one concrete subgroup

In general when doing text-driven analysis, it is not that suitable to use stratified sampling, as we must have some insights and statistical hypothesis about the data and how to form subgroups. Furthermore, the subgroups should be at least somewhat of the same size or contain a minimum of data points. Otherwise a subgroup may contain just one or a few examples which would lead to these definitely being in the stratified sample.

3.

(i) Stop word removal

Advantages:

- Saves space in database
- Saves valuable processing time

- Often are not semantically relevant ("Peter and Susan" -> the "and" does not contain any additional meaning)

Disadvantages:

- Problems arise when searching for phrases that only or mainly contain stop words (e.g. "Take That" or "The Who")
- In some cases the stop words can actually change the whole meaning of a sentence (e.g. "report as the CEO" vs. "report to the CEO")

(ii) Stemming

Advantages:

- Simplifies the phrases and the total amount of words necessary that need to be understood (stem "development" and "developer" to "develop", while maintaining the general meaning)
- Applying text similarity measures on a corpus normalized to word stems is far easier
- Again saves space in database and accelerates processing (i.e. when doing search queries or information retrieval)

Disadvantages:

- Removes semantic information to some extent, as different forms of words contain meaning as well (e.g. "developing" is an action, whereas "development" means a process)
- Chopping of plural endings also removes sometimes very important information

Problem 1-2 PDF Conversion and Regular Expressions

1.

In general, it probably is not a good idea to use regular expressions to parse an HTML file. Even though it is possible to use regular expressions to parse HTML, this only holds for well-formed HTML (context free) with a determined document structure - which is most often not the case. Especially when trying to parse a complete HTML file and understanding the whole structure, regular expressions become much too complex and difficult to maintain. It can quickly happen that too much input is matched, even if it is not supposed to match. On the other hand, it is feasible to use regular expressions to extract little bits of information out of an HTML file, e.g. all links in between `<a>` tags.
So to match simple text patterns using regular expressions is probably the best approach. But to extract information over the whole document - structured or unstructured - usually, it is much easier to just use an HTML parser, which already exist in abundance.

2.

method	qualitative analysis	quantitative analysis (SequenceMatcher.ratio())
online (https://pdftotext.com/de/)	Performed extraordinarily well. Got the whole file structure with separate text blocks right and as far as the considered file goes, did not make any mistake. So it correctly parsed german umlauts and all other characters as well as bullet points. It even managed to interpret word wrappings on line breaks correctly and pieced the word together.	100%
pdfminer	Performs quite decently as it correctly interprets all relevant characters including german umlatus etc. Has some problems with line breaks and vertical text, as it does not convert this into one line.	41.5%

method	qualitative analysis	quantitative analysis (SequenceMatcher.ratio())
PyPDF2	Performed quite well, especially compared to the online method pdf2go and the calibre approach. Correctly parsed german umlauts and got the structure of the pdf file right. One major drawback was, that it interpreted some characters wrongly, e.g. '-', '"' which is especially troublesome when trying to extract phone numbers.	39.1%
pdftotext	Did quite good in converting all the characters correctly from the pdf to txt including german umlauts, bullet points and so on. Correctly transformed vertical text as well. One drawback is that it converts the pdf to txt while maintaining the structure of the original file, which results in txt files where a line is not necessarily a coherent sentence. Can be bad for analyzing context.	19.5%

Comparison based on the conversion of file "FL_SYB_BetriebsaerztlicherDienst_ID8414.pdf".
As a baseline an online tool was used: pdftotext.com.

As one can see in the quantative analysis the pdftotext method achieve only 19.5%, probably due to it maintaining the structure of the original pdf, which is not what we want, actually.
Both the pdfminer and PyPDF2 method did quite similarly well, with pdfminer being slightly better.

But all methods did not fare well compared to the online converte pdftotext, which is why we chose this for the following tasks.

3.

One big problem in general with text conversion and of course this holds true for pdf as well are the various different encodings. This could be observed quite well in the quantitative analysis of task 2, as some conversion methods had problems in
this regard, especially with german umlauts, but also with other characters such as '-' or '"'.

Another problem could be observed in task 2 as well, namely what can happen when a pdf file contains text in different coherent logical blocks, such as multicolumn or some address field inserted arbitrarily. This is difficult to interpret due to the nature of pdf actually being a set of string drawing instructions.

Furthermore it can be difficult to understand what parts of a pdf file actually are supposed to be text and which ones aren't. For example spacing can be meaningful or not, things like arbitrarily designed bullet points are tried to converted to text, even though they don't contain any textual meaning, and so on.

And of course, a pdf file could contain little or no text at all and mainly consist of vector graphics and images. Then the question arises how to interpret these - if at all - or whether maybe some form of OCR should be applied.

5.

It works quite okay with the scanned and ocr interpreted text. But some phone numbers are missed, because they have wrong characters recognized in between them.

Examples:

- "(06221) 43.41 49-0" is not recognized because of the point in between numbers
- "(0 62 21] 4 18 55 58" is not recognized because the closing bracket is

The extraction can be changed to take these into account, but one never can be sure to get all misinterpretations right.

Furthermore, one difficulty is, that a lot of the phone numbers don't have a prefix, because e.g. it is a prefix for all numbers on one page.