Document Structure Recognition

Shan Zhong
University of South Carolina, USA
zhongs@email.sc.edu

Abstract

Electronic documents are a common way to store information. Contracts and data tables from company's public disclosures information, government's orders can be easily accessed and download by internet. These data are valuable, usually people just download and clean them manually, but when we have thousands of websites, documents to download and analysis, then it becomes problematic

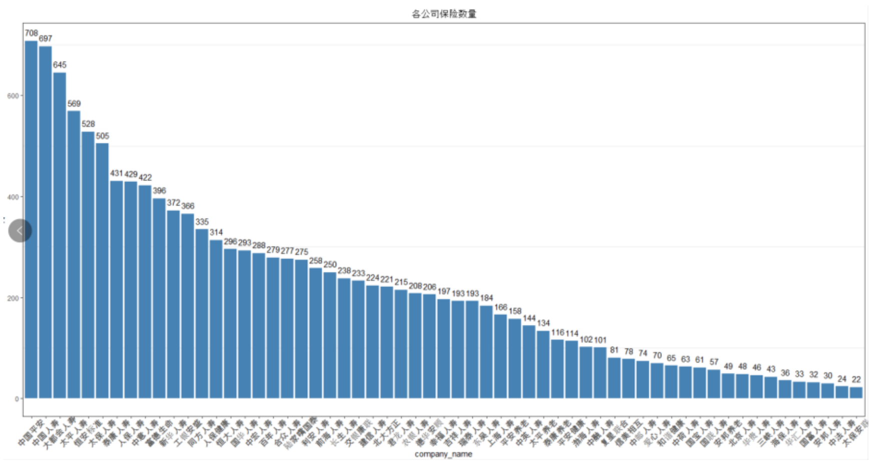
During this summer, I lead a small team with two undergrad students from two top universities of China. They have basic programming skills with also some knowledge in statistical science. We together worked on the document recognition problem.

Goal and purpose

The main problem is currently there is no good way to organize and analyze them in a universal way. As they have a variety of typesetting and formatting (docx, doc, pdf, images, directly posted online….). I also aware that now days there exist tools like Pdf Plumber that can recognize lines and cells in excel, (https://github.com/jsvine/pdfplumber , paper : https://dspace.cc.tut.fi/dpub/bitstream/handle/123456789/21520/Nurminen.pdf?sequence=3), plug in tools like tesseract (https://github.com/tesseract-ocr/) that can do deep learning text recognition, powerful clustering packages like Scikit-learning(https://github.com/scikit-learn/scikit-learn) in python. So being able to utilize and incorporate existing methods that can solve part of the problem and then integrate them into one algorithm that can organize and clean dataset as a human do can be valuable and time-saving, so this is the main purpose for this paper.

Data

Figure1: companies with the number of products they have in y axis.

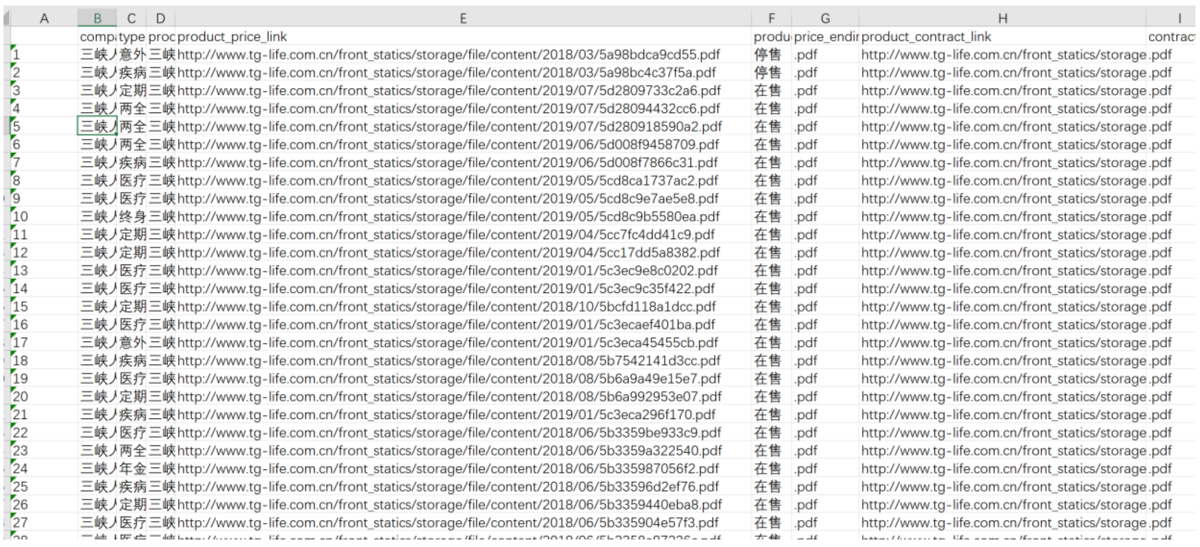


There are 91 insurance companies in china that sells health product, we first get a list for all of them, then find their company website. We used python based scrapping tools Scrapy (https://docs.scrapy.org/en/latest/) to scrape insurance contracts, public disclosure information, insurance coverages description, and premium rate table for their products.

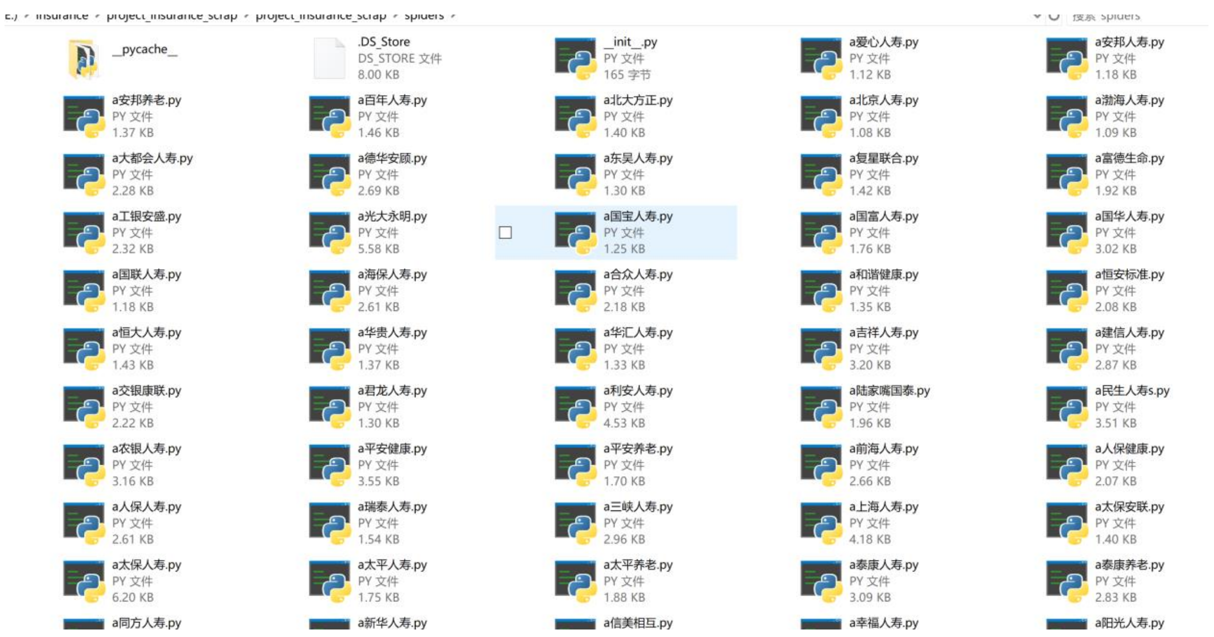
Finally, we found there are around 20,000 products (including those stopped selling) existed. We made a data base for all of them.

How de download the data is we first created a url filename list from the scraping tool’s observation. Some company websites were not that easy to scrap, so we created a virtual machine using windows pro docker to render these websites. We then managed to get a list of all downloadable items from their website. We cleaned these datasets first, then download items according to the links on these lists. Save them locally according to different company different product types and separated them into different folders according to some rule created by our algorithm. Here we created algorithms that can process thousands of files and do operations simultaneously, for example check if successfully download, check if we failed to download and then retry. Some website has software that preventing us downloads lots of material at the same time, so we also adjust algorithm for that.

Figure 2: the url lists



A	B	C	D	E	F	G	H	I
	comp	type	proc	product_price_link	produ	price_endir	product_contract_link	contrac
1	三峡人	意外	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2018/03/5a98bdca9cd55.pdf	停售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
2	三峡人	疾病	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2018/03/5a98bc4c37f5a.pdf	停售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
3	三峡人	定期	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2019/07/5d2809733c2a6.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
4	三峡人	两全	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2019/07/5d28094432cc6.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
5	三峡人	两全	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2019/07/5d280918590a2.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
6	三峡人	两全	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2019/06/5d008f9458709.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
7	三峡人	疾病	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2019/06/5d008f7866c31.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
8	三峡人	医疗	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2019/05/5cd8ca1737ac2.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
9	三峡人	医疗	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2019/05/5cd8c9e7ae5e8.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
10	三峡人	终身	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2019/05/5cd8c9b5580ea.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
11	三峡人	定期	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2019/04/5cc7fc4dd41c9.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
12	三峡人	定期	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2019/04/5cc17dd5a8382.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
13	三峡人	医疗	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2019/01/5c3ec9e8c0202.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
14	三峡人	医疗	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2019/01/5c3ec9c35f422.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
15	三峡人	定期	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2018/10/5bcfd118a1dcc.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
16	三峡人	医疗	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2019/01/5c3ecaef401ba.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
17	三峡人	意外	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2019/01/5c3eca45455cb.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
18	三峡人	疾病	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2018/08/5b7542141d3cc.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
19	三峡人	医疗	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2018/08/5b6a9a49e15e7.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
20	三峡人	定期	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2018/08/5b6a992953e07.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
21	三峡人	疾病	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2019/01/5c3eca296f170.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
22	三峡人	医疗	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2018/06/5b3359be933c9.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
23	三峡人	两全	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2018/06/5b3359a322540.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
24	三峡人	年金	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2018/06/5b335987056f2.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
25	三峡人	疾病	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2018/06/5b33596d2ef76.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
26	三峡人	定期	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2018/06/5b3359440eba8.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf
27	三峡人	医疗	三峡	http://www.tg-life.com.cn/front_statics/storage/file/content/2018/06/5b335904e57f3.pdf	在售	.pdf	http://www.tg-life.com.cn/front_statics/storage	.pdf



E:) › insurance › project_insurance_scrap › project_insurance_scrap › spiders ›   搜索 spiders

__pycache__	.DS_Store DS_STORE 文件 8.00 KB	__init__.py PY 文件 165 字节	a爱心人寿.py PY 文件 1.12 KB	a安邦人寿.py PY 文件 1.18 KB
a安邦养老.py PY 文件 1.37 KB	a百年人寿.py PY 文件 1.46 KB	a北大方正.py PY 文件 1.40 KB	a北京人寿.py PY 文件 1.08 KB	a渤海人寿.py PY 文件 1.09 KB
a大都会人寿.py PY 文件 2.28 KB	a德华安顾.py PY 文件 2.69 KB	a东吴人寿.py PY 文件 1.30 KB	a复星联合.py PY 文件 1.42 KB	a富德生命.py PY 文件 1.92 KB
a工银安盛.py PY 文件 2.32 KB	a光大永明.py PY 文件 5.58 KB	a国宝人寿.py PY 文件 1.25 KB	a国富人寿.py PY 文件 1.76 KB	a国华人寿.py PY 文件 3.02 KB
a国联人寿.py PY 文件 1.18 KB	a海保人寿.py PY 文件 2.61 KB	a合众人寿.py PY 文件 2.18 KB	a和谐健康.py PY 文件 1.35 KB	a恒安标准.py PY 文件 2.08 KB
a恒大人寿.py PY 文件 1.43 KB	a华贵人寿.py PY 文件 1.37 KB	a华汇人寿.py PY 文件 1.33 KB	a吉祥人寿.py PY 文件 3.20 KB	a建信人寿.py PY 文件 2.87 KB
a交银康联.py PY 文件 2.22 KB	a君龙人寿.py PY 文件 1.30 KB	a利安人寿.py PY 文件 4.53 KB	a陆家嘴国泰.py PY 文件 1.96 KB	a民生人寿s.py PY 文件 3.51 KB
a农银人寿.py PY 文件 3.16 KB	a平安健康.py PY 文件 3.55 KB	a平安养老.py PY 文件 1.70 KB	a前海人寿.py PY 文件 2.66 KB	a人保健康.py PY 文件 2.07 KB
a人保人寿.py PY 文件 2.61 KB	a瑞泰人寿.py PY 文件 1.54 KB	a三峡人寿.py PY 文件 2.96 KB	a上海人寿.py PY 文件 4.18 KB	a太保安联.py PY 文件 1.40 KB
a太保人寿.py PY 文件 6.20 KB	a太平人寿.py PY 文件 1.75 KB	a太平养老.py PY 文件 1.88 KB	a泰康人寿.py PY 文件 3.09 KB	a泰康养老.py PY 文件 2.83 KB
a同方人寿.py	a新华人寿.py	a信美相互.py	a幸福人寿.py	a阳光人寿.py

Figure 3, python scrapy code frame with each company, each python file representing a company

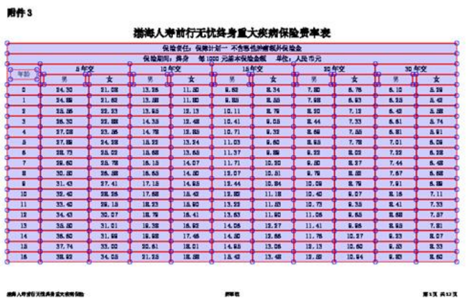
Method

1. Integrating documents into one universal format

From scrapping tools we downloaded thousands of documents to analysis and counted as our sample data. The first thing we need to do is to make them have at least the same format. First, we tried converting all of the documents into a txt format, then we found we lost the structure, front, location of the text, although we can make the text still follow some order and we can still use it to preform analysis

We then transfer all documents into a pdf format, thus information like structures can be remained. We designed several algorithms to detect their structure, read their content, and save them in a more universal way (one dimensional dataset, for example: one column indicates if it is the title or subtitles, another column keeps the whole paragraphs, one column to keep the notes...). Why we want to do so is when just facing insurance documents, there are many similar patterns such as “what are our coverages”, “when we refuse to pay” as documents subtitle. We can then use regular expressions to recognize them.

Figure 4 and 5: detecting lines, and texts.



附件3

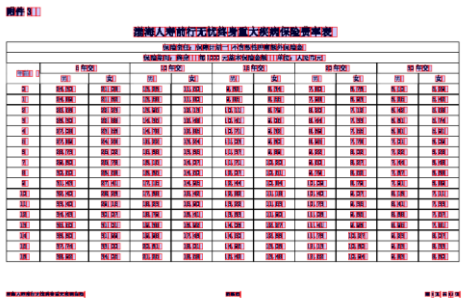
渤海人寿前行无忧终身重大疾病保险费率表

保险责任：保障计划一 不含恶性肿瘤额外保险金

保险期间：终身　每1000元基本保险金额　单位：人民币元

年龄	5年交		10年交		15年交		20年交		30年交	
	男	女	男	女	男	女	男	女	男	女
0	24.30	21.08	13.26	11.50	9.62	8.34	7.80	6.76	6.10	5.29
1	24.89	21.62	13.58	11.80	9.85	8.55	7.98	6.93	6.25	5.42
2	25.56	22.23	13.95	12.13	10.11	8.79	8.20	7.12	6.42	5.58
3	26.30	22.88	14.35	12.48	10.41	9.05	8.44	7.33	6.61	5.74
4	27.08	23.56	14.78	12.85	10.71	9.32	8.69	7.55	6.81	5.91
5	27.89	24.28	15.22	13.24	11.03	9.60	8.95	7.78	7.01	6.09
6	28.73	25.02	15.68	13.65	11.37	9.89	9.22	8.02	7.22	6.28
7	29.60	25.78	16.15	14.07	11.71	10.20	9.50	8.27	7.44	6.48
8	30.50	26.58	16.65	14.50	12.07	10.51	9.79	8.53	7.67	6.68
9	31.43	27.41	17.15	14.95	12.44	10.84	10.09	8.79	7.91	6.89
10	32.40	28.26	17.68	15.42	12.83	11.18	10.40	9.07	8.15	7.11
11	33.40	29.13	18.23	15.90	13.23	11.53	10.73	9.35	8.41	7.33
12	34.43	30.07	18.79	16.41	13.63	11.90	11.06	9.65	8.68	7.57
13	35.50	31.01	19.38	16.92	14.06	12.27	11.41	9.96	8.95	7.81
14	36.60	31.99	19.98	17.46	14.50	12.66	11.76	10.27	9.23	8.07
15	37.74	33.00	20.61	18.01	14.95	13.06	12.13	10.60	9.53	8.33
16	38.92	34.05	21.25	18.58	15.42	13.48	12.52	10.94	9.83	8.60

渤海人寿前行无忧终身重大疾病保险　费率表　第1页 共12页



附件3

渤海人寿前行无忧终身重大疾病保险费率表

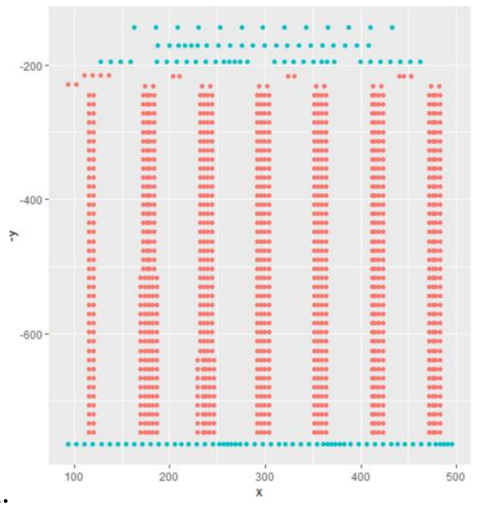
2. **Numeric table and contract documents**

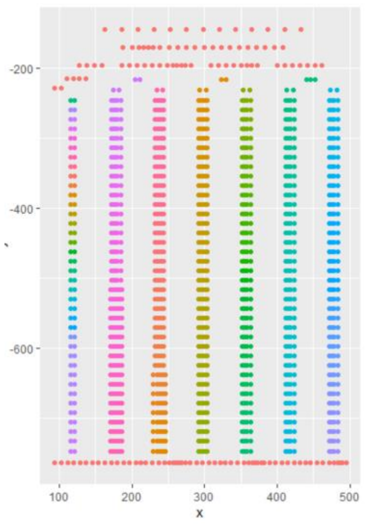
How we determine the structure of text is by detecting the position of every character pieces in the document. Because of the special properties of text, we recognize each of them as a small rectangular. And we also detect horizontal and vertical lines by using python Ocr tools. We do not recognize oblique texts and line, we do not recognize pictures (which we want to focus on later). We get x, y axis position, also width and height of texts. We also get start and end position of lines.

Although there are all kinds of typesetting, but we can still categorize them into two main parts: data tables, and contract document texts. We did not encounter many pictures, and haven’t get a good method to deal with right now.

Data table:

Figure 7 and 8: collapse every character, identify the description part, table parts, and each cell.





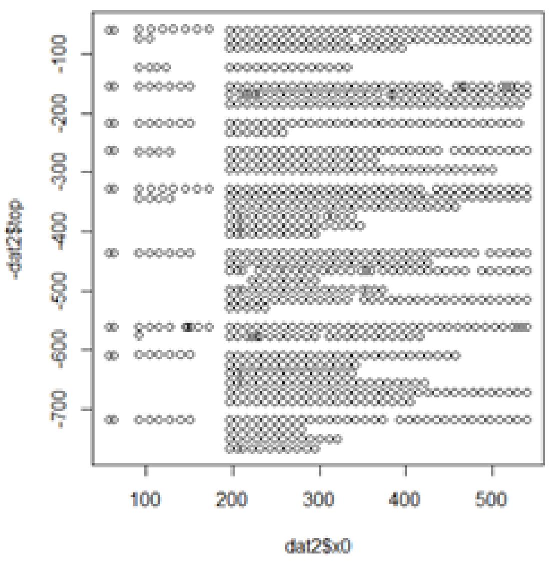
Inside these document files, some of the information are saved as premium tables and mortality tables. These tables have data in two dimensional (parameters are saved in rows and columns relative to each value) that we want to transfer to one dimension (all parameters are saved in columns). Sometimes it is not that easy as there may be several layers of index columns and rows (such as firs layer of columns count gender while second layer of columns count for insurance term periods.

Since we have used Ocr tools to detect the lines in document, we can then get the intersections of horizontal and vertical lines. This allow us to collapse intersections within a tolerance range into one bigger intersection as to represent for a larger cell. We can then combine texts within each cell, align them to each layer of columns and rows to get the other information. For example: check if a cell’s x position is within the x position of a column index. In this way we transferred the data into one dimension.

The difficulty part of the above algorithm is how to implement them in an efficient way. Also, we collected the insurance synonym list so we can save the data in a universal way, which is extremely important when creating one-dimensional dataset to save them.

Word document:

Figure 9 and 10, example of document position recognition

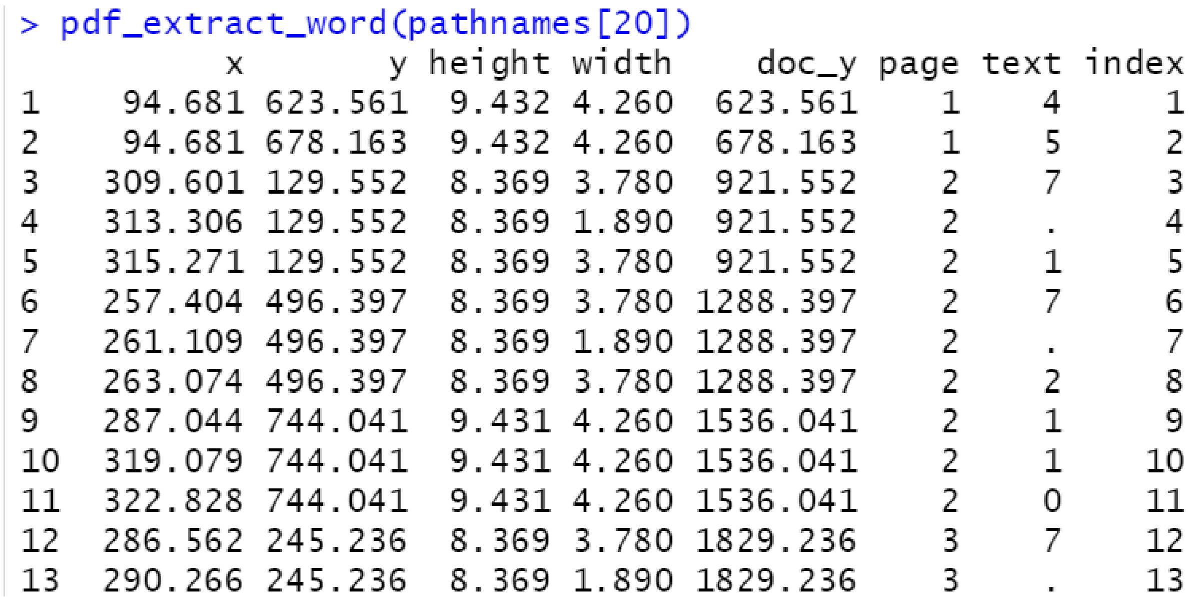


[illegible]

The ultimate final goal for us to analysis these documents is to find the detailed difference in the interpretation of insurance contracts, for different insurance companies and products: In the contracts there do exist similar paragraphs and regulations that allows us to cluster them together. We can then compare which product have more coverage, which product allow us to buy the product easier (insurance products have limitation on who can buy the product, for example those who had cancer before cannot buy cancer insurance, while some product allows in some cases)

For achieving such goals, we first tried a tolerance method: compress all the texts in one dimension, label their x, y position as rectangles, then set the x tolerance and y tolerance of these rectangles. After that, collapse all rectangles within the tolerance range to one bigger rectangle. But this method will usually cause some mistakes and it also not the way that human read and detect text structures. For example, it does not do well on the vertical axis. It also required people to set a tolerance range.

Figure 11, an example of one dimensional data table, with text indicate the original text.



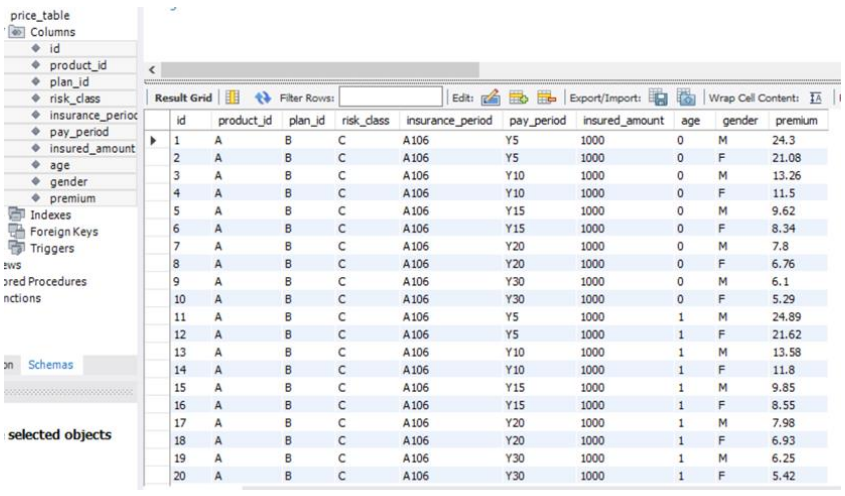
```
> pdf_extract_word(pathnames[20])
          x       y height width    doc_y page text index
1    94.681 623.561  9.432 4.260  623.561    1    4     1
2    94.681 678.163  9.432 4.260  678.163    1    5     2
3   309.601 129.552  8.369 3.780  921.552    2    7     3
4   313.306 129.552  8.369 1.890  921.552    2    .     4
5   315.271 129.552  8.369 3.780  921.552    2    1     5
6   257.404 496.397  8.369 3.780 1288.397    2    7     6
7   261.109 496.397  8.369 1.890 1288.397    2    .     7
8   263.074 496.397  8.369 3.780 1288.397    2    2     8
9   287.044 744.041  9.431 4.260 1536.041    2    1     9
10  319.079 744.041  9.431 4.260 1536.041    2    1    10
11  322.828 744.041  9.431 4.260 1536.041    2    0    11
12  286.562 245.236  8.369 3.780 1829.236    3    7    12
13  290.266 245.236  8.369 1.890 1829.236    3    .    13
```

Then we use a simple k mean clustering method on the x-axis to separate the subtitles and the detailed information besides it. We found just by implementing the easiest clustering algorithm, we can bring a huge increase in the accuracy

We then compare the time the algorithms needed to run in both python and R, and modified computing method to increase algorithm efficiency. We also get a review at the accuracy these algorithms gained.

Finally, we managed to design a new algorithm of identify based on incorporating several existing methods together, and with the input feed in the file path, the data will be transferred and stored in mysql data base.

Figure 12: sample mysql table



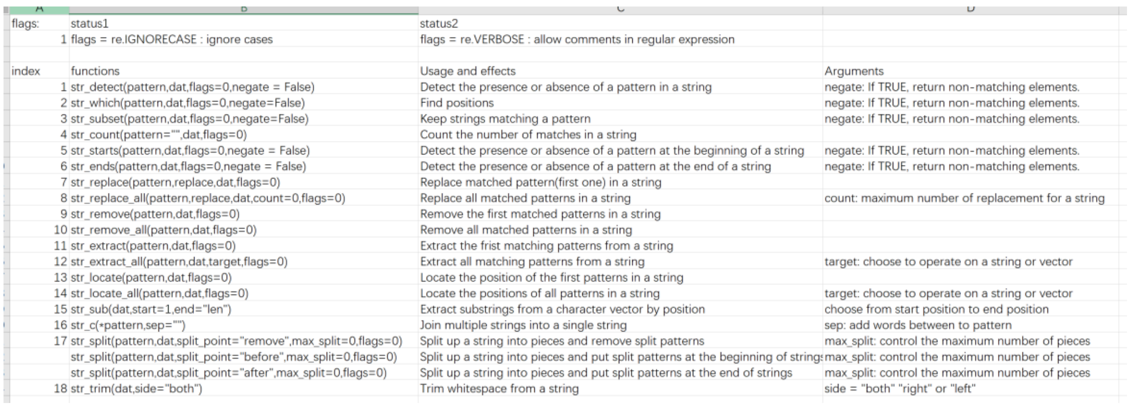
id	product_id	plan_id	risk_class	insurance_period	pay_period	insured_amount	age	gender	premium
1	A	B	C	A106	Y5	1000	0	M	24.3
2	A	B	C	A106	Y5	1000	0	F	21.08
3	A	B	C	A106	Y10	1000	0	M	13.26
4	A	B	C	A106	Y10	1000	0	F	11.5
5	A	B	C	A106	Y15	1000	0	M	9.62
6	A	B	C	A106	Y15	1000	0	F	8.34
7	A	B	C	A106	Y20	1000	0	M	7.8
8	A	B	C	A106	Y20	1000	0	F	6.76
9	A	B	C	A106	Y30	1000	0	M	6.1
10	A	B	C	A106	Y30	1000	0	F	5.29
11	A	B	C	A106	Y5	1000	1	M	24.89
12	A	B	C	A106	Y5	1000	1	F	21.62
13	A	B	C	A106	Y10	1000	1	M	13.58
14	A	B	C	A106	Y10	1000	1	F	11.8
15	A	B	C	A106	Y15	1000	1	M	9.85
16	A	B	C	A106	Y15	1000	1	F	8.55
17	A	B	C	A106	Y20	1000	1	M	7.98
18	A	B	C	A106	Y20	1000	1	F	6.93
19	A	B	C	A106	Y30	1000	1	M	6.25
20	A	B	C	A106	Y30	1000	1	F	5.42

We plan to use information like front size and style to determine whether it is foot notes, title, or descriptive stuffs in the future.

What to do with the data we get

We transfer these word documents into a row of data, indicating each attribute we extract from the contract document. Such as coverage periods, coverage amount…etc. We built up a insurance term dictionary to look up for these words.

Figure 13, example of the word lookup package we built

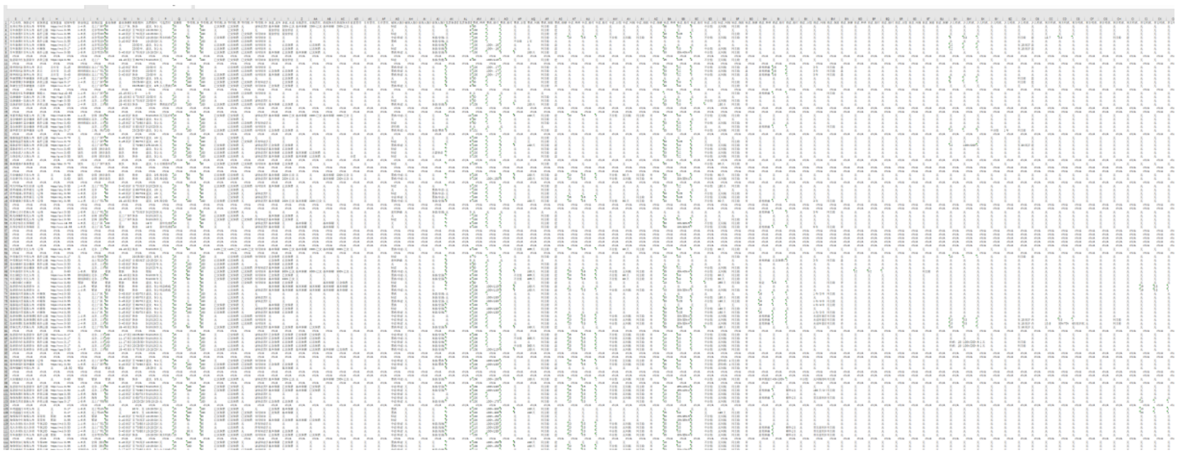


A	B	C	D
flags:	status1	status2	
1	flags = re.IGNORECASE : ignore cases	flags = re.VERBOSE : allow comments in regular expression	
index	functions	Usage and effects	Arguments
1	str_detect(pattern,dat,flags=0,negate = False)	Detect the presence or absence of a pattern in a string	negate: If TRUE, return non-matching elements.
2	str_which(pattern,dat,flags=0,negate=False)	Find positions	negate: If TRUE, return non-matching elements.
3	str_subset(pattern,dat,flags=0,negate=False)	Keep strings matching a pattern	negate: If TRUE, return non-matching elements.
4	str_count(pattern="",dat,flags=0)	Count the number of matches in a string	
5	str_starts(pattern,dat,flags=0,negate = False)	Detect the presence or absence of a pattern at the beginning of a string	negate: If TRUE, return non-matching elements.
6	str_ends(pattern,dat,flags=0,negate = False)	Detect the presence or absence of a pattern at the end of a string	negate: If TRUE, return non-matching elements.
7	str_replace(pattern,replace,dat,flags=0)	Replace matched pattern(first one) in a string	
8	str_replace_all(pattern,replace,dat,count=0,flags=0)	Replace all matched patterns in a string	count: maximum number of replacement for a string
9	str_remove(pattern,dat,flags=0)	Remove the first matched patterns in a string	
10	str_remove_all(pattern,dat,flags=0)	Remove all matched patterns in a string	
11	str_extract(pattern,dat,flags=0)	Extract the frist matching patterns from a string	
12	str_extract_all(pattern,dat,target,flags=0)	Extract all matching patterns from a string	target: choose to operate on a string or vector
13	str_locate(pattern,dat,flags=0)	Locate the position of the first patterns in a string	
14	str_locate_all(pattern,dat,flags=0)	Locate the positions of all patterns in a string	target: choose to operate on a string or vector
15	str_sub(dat,start=1,end="len")	Extract substrings from a character vector by position	choose from start position to end position
16	str_c(*pattern,sep="")	Join multiple strings into a single string	sep: add words between to pattern
17	str_split(pattern,dat,split_point="remove",max_split=0,flags=0)	Split up a string into pieces and remove split patterns	max_split: control the maximum number of pieces
	str_split(pattern,dat,split_point="before",max_split=0,flags=0)	Split up a string into pieces and put split patterns at the beginning of strings	max_split: control the maximum number of pieces
	str_split(pattern,dat,split_point="after",max_split=0,flags=0)	Split up a string into pieces and put split patterns at the end of strings	max_split: control the maximum number of pieces
18	str_trim(dat,side="both")	Trim whitespace from a string	side = "both" "right" or "left"

Also, for conveniently look for combination of words in python. We designed our customized regular expression lookup functions. We mimic the R Stringr package and added several additional features.

We matched the illness definition by the insurance contract, to check if there are overlaps, and check the percentage covered by each different insurance product. We can then put these attributes into the column for our later analysis.

Figure 14: we have a lot of variables, there are around 100.



Our dependent variable will of course be the price, we built a contract valuation regression model to predict the actuarial value of contracts sold by each different company. As oppose to the pure actuarial model which was based on the modification of mortality table. We compared the result for both model, and present to our customers.

We used the 2010 china Critical illness life table and the mortality table in building the pure actuarial model. We found using regression model, it is much earlier to explain them to customers. We can directly figure out why one product is more expensive when doing comparison, and find out the most valuable products.

Conclusion

This paper is mostly about the implementation part, I still need to work on the algorithm to really discover something totally new.