

Personal Information

Name: **Roderick Majoor**

StudentID: **12852724**

Email: roderick.majoor@student.uva.nl

Submitted on: **22.03.2024**

Data Context

The dataset used in this study consists of pages from the collection of ledgers from the Bank of Amsterdam. The full collection can be found on the website of The Amsterdam City Archives and consists of approximately half a million pages. In the series of ledgers, the current accounts of the customers were maintained. Each customer had one or more pages, with smaller traders having a portion of a page. An alphabetical list of the traders and the page of their current account can be found in the index. The ledgers kept track of the amounts that were debited and credited from and to a customer. The ledger pages consist of several columns containing information such as the date, account holder name, account number and amount debited/credited.

Data Description

The total dataset consists of approximately half a million pages from The ledgers of the Bank of Amsterdam (Amsterdamse Wisselbank) from between 1650-1800. The pages follow a tabular structure where on each page, the debit/credit account of a client is kept track of. The columns of both the debit and credit table show a date, account name, guldens, stuivers, and penningen (currency at the time). The ground truth annotated set consist of 68 pages

Intro

Currently, a tool called Loghi is used to perform handwritten text recognition on these pages. This tool has its flaws though, and in this thesis we are trying to determine what the major errors are and how we could possibly solve them.

From the analysis show below we find the following:

1. False positives detected (typically due to marks / stripes on the image that are recognized as text)
2. Numbers are not detected as numbers but rather as text / other characters
3. Words/numbers are merged or separated (i.e. 16 becomes 1, 6 or 2, 8 becomes 28 etc.)
4. Numbers not correctly recognized (i.e. 16 becomes 18 etc.)
5. Output not in the correct layout

Some of the code used is quite long, so instead of putting everything in this notebook, we will just import it from their scripts. The scripts can be found on the GitHub:

<https://github.com/roderickmajoer/Thesis/tree/main>

```
In [1]: # Imports
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import xml.etree.ElementTree as ET
import jiwer
import cv2

from compare_GT_data_textlines import main
```

Data Loading

```
In [2]: # Path to the folder containing the images
data_dir = "/home/roderickmajoer/Desktop/Master/Thesis/GT_data"

# List to store paths of all image files
image_paths = []

# Iterate through the folder structure
for root, dirs, files in os.walk(data_dir):
    for file in files:
        # Check if the file is an image
        if file.endswith(".jpg") or file.endswith(".png"):
            # Construct the full path to the image file
            image_path = os.path.join(root, file)
            # Append the image path to the list
            image_paths.append(image_path)
```

Analysis 1: Visualizing Data

Here we show a couple of the images in our dataset. This already shows us a couple of things. First of all, the sizes of the images are not always the same. The first 3 images are larger than image 4 and 5. The first 3 images also seem to have a more 'neat' layout where text does not flow over into other text. The layout on image 4 and 5 seems a bit more

chaotic. Other noticeable things are stains on some of the pages and also stripes/marks that go through some of the text which may cause problems when analysing it.

```
In [3]: # Set the number of images to visualize
num_images_to_visualize = 5

# Visualize a sample of the images
for i in range(num_images_to_visualize):
    # Read the image using matplotlib
    image = mpimg.imread(image_paths[i])
    # Plot the image
    plt.figure(figsize=(8, 8))
    plt.imshow(image)
    plt.axis('off') # Turn off axis
    plt.title(f"Image {i+1}") # Set title
    plt.show()
```

Image 1

Cedula no. 1			
10	227	J. D. de la C. 10000	228
11	228	J. D. de la C. 10000	229
12	229	J. D. de la C. 10000	230
13	230	J. D. de la C. 10000	231
14	231	J. D. de la C. 10000	232
15	232	J. D. de la C. 10000	233
16	233	J. D. de la C. 10000	234
17	234	J. D. de la C. 10000	235
18	235	J. D. de la C. 10000	236
19	236	J. D. de la C. 10000	237
20	237	J. D. de la C. 10000	238
21	238	J. D. de la C. 10000	239
22	239	J. D. de la C. 10000	240
23	240	J. D. de la C. 10000	241
24	241	J. D. de la C. 10000	242
25	242	J. D. de la C. 10000	243
26	243	J. D. de la C. 10000	244
27	244	J. D. de la C. 10000	245
28	245	J. D. de la C. 10000	246
29	246	J. D. de la C. 10000	247
30	247	J. D. de la C. 10000	248
31	248	J. D. de la C. 10000	249
32	249	J. D. de la C. 10000	250
33	250	J. D. de la C. 10000	251
34	251	J. D. de la C. 10000	252
35	252	J. D. de la C. 10000	253
36	253	J. D. de la C. 10000	254
37	254	J. D. de la C. 10000	255
38	255	J. D. de la C. 10000	256
39	256	J. D. de la C. 10000	257
40	257	J. D. de la C. 10000	258
41	258	J. D. de la C. 10000	259
42	259	J. D. de la C. 10000	260
43	260	J. D. de la C. 10000	261
44	261	J. D. de la C. 10000	262
45	262	J. D. de la C. 10000	263
46	263	J. D. de la C. 10000	264
47	264	J. D. de la C. 10000	265
48	265	J. D. de la C. 10000	266
49	266	J. D. de la C. 10000	267
50	267	J. D. de la C. 10000	268
51	268	J. D. de la C. 10000	269
52	269	J. D. de la C. 10000	270
53	270	J. D. de la C. 10000	271
54	271	J. D. de la C. 10000	272
55	272	J. D. de la C. 10000	273
56	273	J. D. de la C. 10000	274
57	274	J. D. de la C. 10000	275
58	275	J. D. de la C. 10000	276
59	276	J. D. de la C. 10000	277
60	277	J. D. de la C. 10000	278
61	278	J. D. de la C. 10000	279
62	279	J. D. de la C. 10000	280
63	280	J. D. de la C. 10000	281
64	281	J. D. de la C. 10000	282
65	282	J. D. de la C. 10000	283
66	283	J. D. de la C. 10000	284
67	284	J. D. de la C. 10000	285
68	285	J. D. de la C. 10000	286
69	286	J. D. de la C. 10000	287
70	287	J. D. de la C. 10000	288
71	288	J. D. de la C. 10000	289
72	289	J. D. de la C. 10000	290
73	290	J. D. de la C. 10000	291
74	291	J. D. de la C. 10000	292
75	292	J. D. de la C. 10000	293
76	293	J. D. de la C. 10000	294
77	294	J. D. de la C. 10000	295
78	295	J. D. de la C. 10000	296
79	296	J. D. de la C. 10000	297
80	297	J. D. de la C. 10000	298
81	298	J. D. de la C. 10000	299
82	299	J. D. de la C. 10000	300
83	300	J. D. de la C. 10000	301
84	301	J. D. de la C. 10000	302
85	302	J. D. de la C. 10000	303
86	303	J. D. de la C. 10000	304
87	304	J. D. de la C. 10000	305
88	305	J. D. de la C. 10000	306
89	306	J. D. de la C. 10000	307
90	307	J. D. de la C. 10000	308
91	308	J. D. de la C. 10000	309
92	309	J. D. de la C. 10000	310
93	310	J. D. de la C. 10000	311
94	311	J. D. de la C. 10000	312
95	312	J. D. de la C. 10000	313
96	313	J. D. de la C. 10000	314
97	314	J. D. de la C. 10000	315
98	315	J. D. de la C. 10000	316
99	316	J. D. de la C. 10000	317
100	317	J. D. de la C. 10000	318
101	318	J. D. de la C. 10000	319
102	319	J. D. de la C. 10000	320
103	320	J. D. de la C. 10000	321
104	321	J. D. de la C. 10000	322
105	322	J. D. de la C. 10000	323
106	323	J. D. de la C. 10000	324
107	324	J. D. de la C. 10000	325
108	325	J. D. de la C. 10000	326
109	326	J. D. de la C. 10000	327
110	327	J. D. de la C. 10000	328
111	328	J. D. de la C. 10000	329
112	329	J. D. de la C. 10000	330
113	330	J. D. de la C. 10000	331
114	331	J. D. de la C. 10000	332
115	332	J. D. de la C. 10000	333
116	333	J. D. de la C. 10000	334
117	334	J. D. de la C. 10000	335
118	335	J. D. de la C. 10000	336
119	336	J. D. de la C. 10000	337
120	337	J. D. de la C. 10000	338
121	338	J. D. de la C. 10000	339
122	339	J. D. de la C. 10000	340
123	340	J. D. de la C. 10000	341
124	341	J. D. de la C. 10000	342
125	342	J. D. de la C. 10000	343
126	343	J. D. de la C. 10000	344
127	344	J. D. de la C. 10000	345
128	345	J. D. de la C. 10000	346
129	346	J. D. de la C. 10000	347
130	347	J. D. de la C. 10000	348
131	348	J. D. de la C. 10000	349
132	349	J. D. de la C. 10000	350
133	350	J. D. de la C. 10000	351
134	351	J. D. de la C. 10000	352
135	352	J. D. de la C. 10000	353
136	353	J. D. de la C. 10000	354
137	354	J. D. de la C. 10000	355
138	355	J. D. de la C. 10000	356
139	356	J. D. de la C. 10000	357
140	357	J. D. de la C. 10000	358
141	358	J. D. de la C. 10000	359
142	359	J. D. de la C. 10000	360
143	360	J. D. de la C. 10000	361
144	361	J. D. de la C. 10000	362
145	362	J. D. de la C. 10000	363
146	363	J. D. de la C. 10000	364
147	364	J. D. de la C. 10000	365
148	365	J. D. de la C. 10000	366
149	366	J. D. de la C. 10000	367
150	367	J. D. de la C. 10000	368
151	368	J. D. de la C. 10000	369
152	369	J. D. de la C. 10000	370
153	370	J. D. de la C. 10000	371
154	371	J. D. de la C. 10000	372
155	372	J. D. de la C. 10000	373
156	373	J. D. de la C. 10000	374
157	374	J. D. de la C. 10000	375
158	375	J. D. de la C. 10000	376
159	376	J. D. de la C. 10000	377
160	377	J. D. de la C. 10000	378
161	378	J. D. de la C. 10000	379
162	379	J. D. de la C. 10000	380
163	380	J. D. de la C. 10000	381
164	381	J. D. de la C. 10000	382
165	382	J. D. de la C. 10000	383
166	383	J. D. de la C. 10000	384
167	384	J. D. de la C. 10000	385
168	385	J. D. de la C. 10000	386
169	386	J. D. de la C. 10000	387
170	387	J. D. de la C. 10000	388
171	388	J. D. de la C. 10000	389
172	389	J. D. de la C. 10000	390
173	390	J. D. de la C. 10000	391
174	391	J. D. de la C. 10000	392
175	392	J. D. de la C. 10000	393
176	393	J. D. de la C. 10000	394
177	394	J. D. de la C. 10000	395
178	395	J. D. de la C. 10000	396
179	396	J. D. de la C. 10000	397
180	397	J. D. de la C. 10000	398
181	398	J. D. de la C. 10000	399
182	399	J. D. de la C. 10000	400
183	400	J. D. de la C. 10000	401
184	401	J. D. de la C. 10000	402
185	402	J. D. de la C. 10000	403
186	403	J. D. de la C. 10000	404
187	404	J. D. de la C. 10000	405
188	405	J. D. de la C. 10000	406
189	406	J. D. de la C. 10000	407
190	407	J. D. de la C. 10000	408
191	408	J. D. de la C. 10000	409
192	409	J. D. de la C. 10000	410
193	410	J. D. de la C. 10000	411
194	411	J. D. de la C. 10000	412
195	412	J. D. de la C. 10000	413
196	413	J. D. de la C. 10000	414
197	414	J. D. de la C. 10000	415
198	415	J. D. de la C. 10000	416
199	416	J. D. de la C. 10000	417
200	417	J. D. de la C. 10000	418
201	418	J. D. de la C. 10000	419
202	419	J. D. de la C. 10000	420
203	420	J. D. de la C. 10000	421
204	421	J. D. de la C. 10000	422
205	422	J. D. de la C. 10000	423
206	423	J. D. de la C. 10000	424
207	424	J. D. de la C. 10000	425
208	425	J. D. de la C. 10000	426
209	426	J. D. de la C. 10000	427
210	427	J. D. de la C. 10000	428
211	428	J. D. de la C. 10000	429
212	429	J. D. de la C. 10000	430
213	430	J. D. de la C. 10000	431
214	431	J. D. de la C. 10000	432
215	432	J. D. de la C. 10000	433
216	433	J. D. de la C. 10000	434
217	434	J. D. de la C. 10000	435
218	435	J. D. de la C. 10000	436
219	436	J. D. de la C. 10000	437
220	437	J. D. de la C. 10000	438
221	438	J. D. de la C. 10000	439
222	439	J. D. de la C. 10000	440
223	440	J. D. de la C. 10000	441
224	441	J. D. de la C. 10000	442
225	442	J. D. de la C. 10000	443
226	443	J. D. de la C. 10000	444
227	444	J. D. de la C. 10000	445
228	445	J. D. de la C. 10000	446
229	446	J. D. de la C. 10000	447
230	447	J. D. de la C. 10000	448
231	448	J. D. de la C. 10000	449
232	449	J. D. de la C. 10000	450
233	450	J. D. de la C. 10000	451
234	451	J. D. de la C. 10000	452
235	452	J. D. de la C. 10000	453
236	453	J. D. de la C. 10000	454
237	454	J. D. de la C. 10000	455
238	455	J. D. de la C. 10000	456
239	456	J. D. de la C. 10000	457
240	457	J. D. de la C. 10000	458
241	458	J. D. de la C. 10000	459
242	459	J. D. de la C. 10000	460
243	460	J. D. de la C. 10000	461
244	461	J. D. de la C. 10000	462
245	462	J. D. de la C. 10000	463
246	463	J. D. de la C. 10000	464

Image 2

Amsterdam Anne 1763		Amsterdam Anne 1763			
Groot Secret		Klein Secret			
<i>Sekretarie van alle het grote en kleine gevonden te Haer, Zee & der Stadt Amsterdam op den Januarij d. 1763.</i>					
<i>Stad en Provintialen Vast. K. S. Secret op den Januarij 1763.</i>					
Fols. 1 tot 3	148	Goud-Ducaten	1600,-		
4 - C	35	Goud-Ducaten	1200,-		
5 - D	242	Goud-Ducaten	1000,-		
4 - E	60	Goud-Ducaten	600,-		
4 - F	700	Goud-Ducaten	480,-		
5 - G	200	Muzzen	2000,-		
5 - H	120	Dito	120,-		
6 - I	120	Dito	120,-		
6 - K	120	Dito	120,-		
7 - L	120	Dito	120,-		
8 - M	60	Verlorenen	1200,-		
8 - N	600	Nieuwe Spanjaarders	480,-		
9 - O	120	Muzzen	1200,-		
11 - P	500	Amsterdamsche Duratoren	1000,-		
12 - AA	300	Muzzen	1200,-		
15 - BB	300	Dito	600,-		
16 - DD	300	Dito	600,-		
17 - EE	300	Dito	600,-		
18 - FF	300	Dito	600,-		
19 - GG	300	Dito	600,-		
18 - HH	300	Dito	600,-		
19 - JJ	300	Dito	600,-		
19 - KK	300	Dito	600,-		
20 - LL	300	Dito	600,-		
20 - MM	300	Dito	600,-		
21 - NN	300	Dito	600,-		
22 - OO	300	Dito	600,-		
23 - PP	300	Dito	600,-		
24 - QQ	2400	Belassen Kapel 11	6280.000		
32 -	1137	Nieuwe Franse Croon Kapel 12	2501.400		
34 -			22568.914 £		

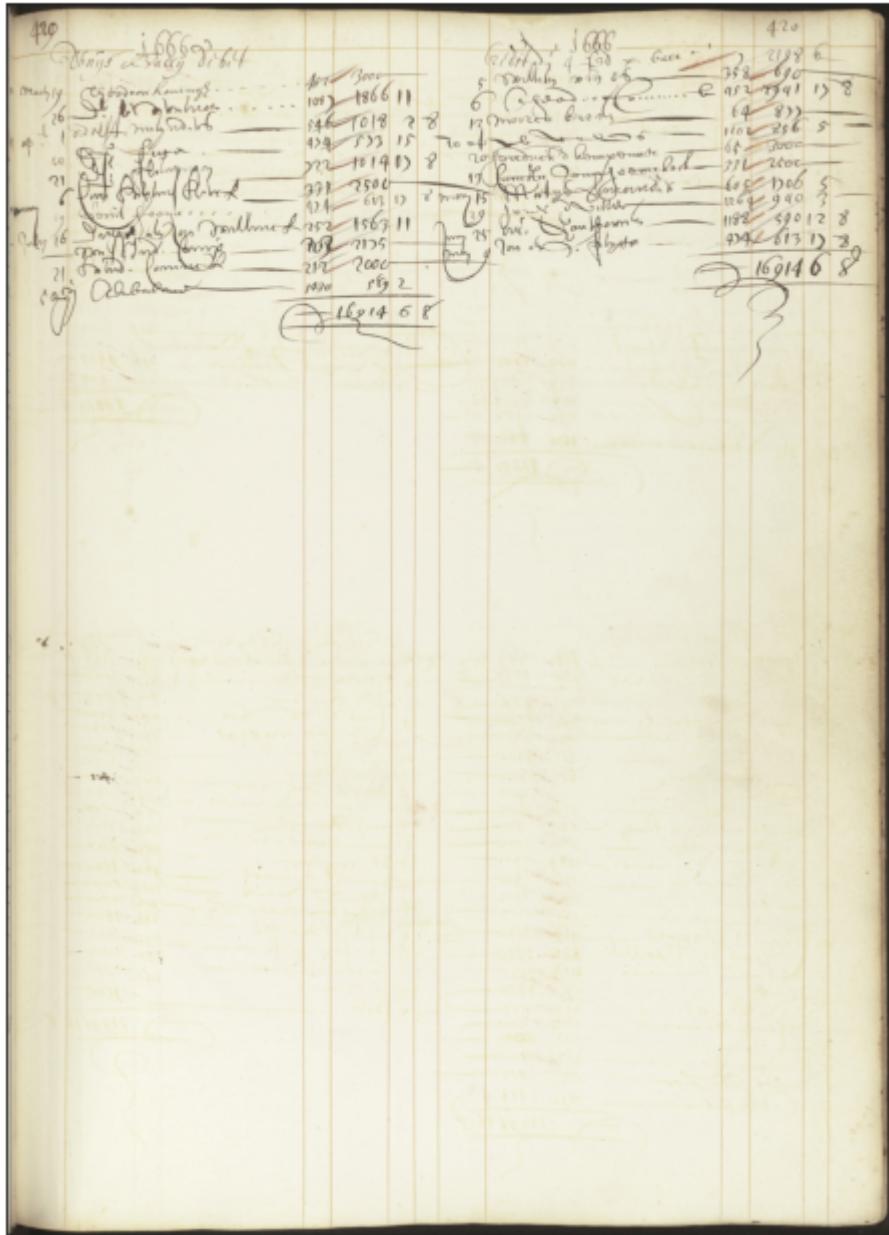
Image 3

AenWyzing der Winsten op 28 Januarij 1763.		AenWyzing der Winsten op 28 Januarij 1763.	
fol. 1	2	op de Nieuwe Croon	374,-
41	3	Goud-Ducaten a 500 fl.	1224.11,-
43	4	goud-Ducaten	1,-
46	5	Goud-Ducaten a 1000 fl.	623.50,-
48	6	Goud-Ducaten	1176.13,-
51	7	Goud-Ducaten a 1000 fl.	2263.12,-
56	8	Goud-Ducaten	404.50,-
64	11	De Gouden	1033,-
78	13	De Muzzen	4396,-
86	14	Goud-Ducaten	105.15,-
28	15	Spanische Croon	17.63.10,-
95	16	Verlorenen	11,-
99	18	Goud-Ducaten	143.6,-
100	17	Goud-Ducaten	162.6,-
102	17	Goud-Ducaten	176.10,-
107	17	Goud-Ducaten	2591.12,-
			181757.61
			2663.4,-
		Van de Oostenrijksche Compagnie van Prag gold van de Rekenh. N.	426.0
		Van Prag gold van de Rekenh. N.	426.0
		Van Prag gold van de Rekenh. N.	426.0
		Aen de Schiedt Gold van Januarij 1763	426.0
			145725.10.0
		Affordde 1000 reken van der Obligation op de Stelling	250.0
		en Aen de Winsten op de Rekenh. N.	250.0
		Vergoede Januarij 1763 250.00	250.0
		af Aen de Stelling	250.00
			250.00
		Tijverdienst	17240.11

Image 4

	620	166		620
1. <i>François Desirée leottt</i>			6/1/18 192 19 8	
2. <i>Proprietary aluminum</i>	61 140		36 1005 78	
3. <i>Stainless steel</i>	10 725		58 1000	
4. <i>Aluminum frame for bicycle</i>	450 125		54 1000	
5. <i>aluminum</i>	860 120		51 100	
6. <i>aluminum</i>	107 100		24 100 900	
7. <i>Metal for plates</i>	200 1000		23 1000	
8. <i>aluminum</i>	900 600		22 1000	
9. <i>aluminum</i>	1000 350		10 1000	
10. <i>aluminum</i>	1000 200		11 1000	
11. <i>aluminum</i>	1000 100		12 1000	
12. <i>aluminum</i>	1000 100		13 1000	
13. <i>aluminum</i>	1000 100		14 1000	
14. <i>aluminum</i>	1000 100		15 1000	
15. <i>aluminum</i>	1000 100		16 1000	
16. <i>aluminum</i>	1000 100		17 1000	
17. <i>aluminum</i>	1000 100		18 1000	
18. <i>aluminum</i>	1000 100		19 1000	
19. <i>aluminum</i>	1000 100		20 1000	
20. <i>aluminum</i>	1000 100		21 1000	
21. <i>aluminum</i>	1000 100		22 1000	
22. <i>aluminum</i>	1000 100		23 1000	
23. <i>aluminum</i>	1000 100		24 1000	
24. <i>aluminum</i>	1000 100		25 1000	
25. <i>aluminum</i>	1000 100		26 1000	
26. <i>aluminum</i>	1000 100		27 1000	
27. <i>aluminum</i>	1000 100		28 1000	
28. <i>aluminum</i>	1000 100		29 1000	
29. <i>aluminum</i>	1000 100		30 1000	
30. <i>aluminum</i>	1000 100		31 1000	
31. <i>aluminum</i>	1000 100		32 1000	
32. <i>aluminum</i>	1000 100		33 1000	
33. <i>aluminum</i>	1000 100		34 1000	
34. <i>aluminum</i>	1000 100		35 1000	
35. <i>aluminum</i>	1000 100		36 1000	
36. <i>aluminum</i>	1000 100		37 1000	
37. <i>aluminum</i>	1000 100		38 1000	
38. <i>aluminum</i>	1000 100		39 1000	
39. <i>aluminum</i>	1000 100		40 1000	
40. <i>aluminum</i>	1000 100		41 1000	
41. <i>aluminum</i>	1000 100		42 1000	
42. <i>aluminum</i>	1000 100		43 1000	
43. <i>aluminum</i>	1000 100		44 1000	
44. <i>aluminum</i>	1000 100		45 1000	
45. <i>aluminum</i>	1000 100		46 1000	
46. <i>aluminum</i>	1000 100		47 1000	
47. <i>aluminum</i>	1000 100		48 1000	
48. <i>aluminum</i>	1000 100		49 1000	
49. <i>aluminum</i>	1000 100		50 1000	
50. <i>aluminum</i>	1000 100		51 1000	
51. <i>aluminum</i>	1000 100		52 1000	
52. <i>aluminum</i>	1000 100		53 1000	
53. <i>aluminum</i>	1000 100		54 1000	
54. <i>aluminum</i>	1000 100		55 1000	
55. <i>aluminum</i>	1000 100		56 1000	
56. <i>aluminum</i>	1000 100		57 1000	
57. <i>aluminum</i>	1000 100		58 1000	
58. <i>aluminum</i>	1000 100		59 1000	
59. <i>aluminum</i>	1000 100		60 1000	
60. <i>aluminum</i>	1000 100		61 1000	
61. <i>aluminum</i>	1000 100		62 1000	
62. <i>aluminum</i>	1000 100		63 1000	
63. <i>aluminum</i>	1000 100		64 1000	
64. <i>aluminum</i>	1000 100		65 1000	
65. <i>aluminum</i>	1000 100		66 1000	
66. <i>aluminum</i>	1000 100		67 1000	
67. <i>aluminum</i>	1000 100		68 1000	
68. <i>aluminum</i>	1000 100		69 1000	
69. <i>aluminum</i>	1000 100		70 1000	
70. <i>aluminum</i>	1000 100		71 1000	
71. <i>aluminum</i>	1000 100		72 1000	
72. <i>aluminum</i>	1000 100		73 1000	
73. <i>aluminum</i>	1000 100		74 1000	
74. <i>aluminum</i>	1000 100		75 1000	
75. <i>aluminum</i>	1000 100		76 1000	
76. <i>aluminum</i>	1000 100		77 1000	
77. <i>aluminum</i>	1000 100		78 1000	
78. <i>aluminum</i>	1000 100		79 1000	
79. <i>aluminum</i>	1000 100		80 1000	
80. <i>aluminum</i>	1000 100		81 1000	
81. <i>aluminum</i>	1000 100		82 1000	
82. <i>aluminum</i>	1000 100		83 1000	
83. <i>aluminum</i>	1000 100		84 1000	
84. <i>aluminum</i>	1000 100		85 1000	
85. <i>aluminum</i>	1000 100		86 1000	
86. <i>aluminum</i>	1000 100		87 1000	
87. <i>aluminum</i>	1000 100		88 1000	
88. <i>aluminum</i>	1000 100		89 1000	
89. <i>aluminum</i>	1000 100		90 1000	
90. <i>aluminum</i>	1000 100		91 1000	
91. <i>aluminum</i>	1000 100		92 1000	
92. <i>aluminum</i>	1000 100		93 1000	
93. <i>aluminum</i>	1000 100		94 1000	
94. <i>aluminum</i>	1000 100		95 1000	
95. <i>aluminum</i>	1000 100		96 1000	
96. <i>aluminum</i>	1000 100		97 1000	
97. <i>aluminum</i>	1000 100		98 1000	
98. <i>aluminum</i>	1000 100		99 1000	
99. <i>aluminum</i>	1000 100		100 1000	
100. <i>aluminum</i>	1000 100		101 1000	
101. <i>aluminum</i>	1000 100		102 1000	
102. <i>aluminum</i>	1000 100		103 1000	
103. <i>aluminum</i>	1000 100		104 1000	
104. <i>aluminum</i>	1000 100		105 1000	
105. <i>aluminum</i>	1000 100		106 1000	
106. <i>aluminum</i>	1000 100		107 1000	
107. <i>aluminum</i>	1000 100		108 1000	
108. <i>aluminum</i>	1000 100		109 1000	
109. <i>aluminum</i>	1000 100		110 1000	
110. <i>aluminum</i>	1000 100		111 1000	
111. <i>aluminum</i>	1000 100		112 1000	
112. <i>aluminum</i>	1000 100		113 1000	
113. <i>aluminum</i>	1000 100		114 1000	
114. <i>aluminum</i>	1000 100		115 1000	
115. <i>aluminum</i>	1000 100		116 1000	
116. <i>aluminum</i>	1000 100		117 1000	
117. <i>aluminum</i>	1000 100		118 1000	
118. <i>aluminum</i>	1000 100		119 1000	
119. <i>aluminum</i>	1000 100		120 1000	
120. <i>aluminum</i>	1000 100		121 1000	
121. <i>aluminum</i>	1000 100		122 1000	
122. <i>aluminum</i>	1000 100		123 1000	
123. <i>aluminum</i>	1000 100		124 1000	
124. <i>aluminum</i>	1000 100		125 1000	
125. <i>aluminum</i>	1000 100		126 1000	
126. <i>aluminum</i>	1000 100		127 1000	
127. <i>aluminum</i>	1000 100		128 1000	
128. <i>aluminum</i>	1000 100		129 1000	
129. <i>aluminum</i>	1000 100		130 1000	
130. <i>aluminum</i>	1000 100		131 1000	
131. <i>aluminum</i>	1000 100		132 1000	
132. <i>aluminum</i>	1000 100		133 1000	
133. <i>aluminum</i>	1000 100		134 1000	
134. <i>aluminum</i>	1000 100		135 1000	
135. <i>aluminum</i>	1000 100		136 1000	
136. <i>aluminum</i>	1000 100		137 1000	
137. <i>aluminum</i>	1000 100		138 1000	
138. <i>aluminum</i>	1000 100		139 1000	
139. <i>aluminum</i>	1000 100		140 1000	
140. <i>aluminum</i>	1000 100		141 1000	
141. <i>aluminum</i>	1000 100		142 1000	
142. <i>aluminum</i>	1000 100		143 1000	
143. <i>aluminum</i>	1000 100		144 1000	
144. <i>aluminum</i>	1000 100		145 1000	
145. <i>aluminum</i>	1000 100		146 1000	
146. <i>aluminum</i>	1000 100		147 1000	
147. <i>aluminum</i>	1000 100		148 1000	
148. <i>aluminum</i>	1000 100		149 1000	
149. <i>aluminum</i>	1000 100		150 1000	
150. <i>aluminum</i>	1000 100		151 1000	
151. <i>aluminum</i>	1000 100		152 1000	
152. <i>aluminum</i>	1000 100		153 1000	
153. <i>aluminum</i>	1000 100		154 1000	
154. <i>aluminum</i>	1000 100		155 1000	
155. <i>aluminum</i>	1000 100		156 1000	
156. <i>aluminum</i>	1000 100		157 1000	
157. <i>aluminum</i>	1000 100		158 1000	
158. <i>aluminum</i>	1000 100		159 1000	
159. <i>aluminum</i>	1000 100		160 1000	
160. <i>aluminum</i>	1000 100		161 1000	
161. <i>aluminum</i>	1000 100		162 1000	
162. <i>aluminum</i>	1000 100		163 1000	
163. <i>aluminum</i>	1000 100		164 1000	
164. <i>aluminum</i>	1000 100		165 1000	
165. <i>aluminum</i>	1000 100		166 1000	
166. <i>aluminum</i>	1000 100		167 1000	
167. <i>aluminum</i>	1000 100		168 1000	
168. <i>aluminum</i>	1000 100		169 1000	
169. <i>aluminum</i>	1000 100		170 1000	
170. <i>aluminum</i>	1000 100		171 1000	
171. <i>aluminum</i>	1000 100		172 1000	
172. <i>aluminum</i>	1000 100		173 1000	
173. <i>aluminum</i>	1000 100		174 1000	
174. <i>aluminum</i>	1000 100		175 1000	
175. <i>aluminum</i>	1000 100		176 1000	
176. <i>aluminum</i>	1000 100		177 1000	
177. <i>aluminum</i>	1000 100		178 1000	
178. <i>aluminum</i>	1000 100		179 1000	
179. <i>aluminum</i>	1000 100		180 1000	
180. <i>aluminum</i>	1000 100		181 1000	
181. <i>aluminum</i>	1000 100		182 1000	
182. <i>aluminum</i>	1000 100		183 1000	
183. <i>aluminum</i>	1000 100		184 1000	
184. <i>aluminum</i>	1000 100		185 1000	
185. <i>aluminum</i>	1000 100		186 1000	
186. <i>aluminum</i>	1000 100		187 1000	
187. <i>aluminum</i>	1000 100		188 1000	
188. <i>aluminum</i>	1000 100		189 1000	
189. <i>aluminum</i>	1000 100		190 1000	
190. <i>aluminum</i>	1000 100		191 1000	
191. <i>aluminum</i>	1000 100		192 1000	
192. <i>aluminum</i>	1000 100		193 1000	
193. <i>aluminum</i>	1000 100		194 1000	
194. <i>aluminum</i>	1000 100		195 1000	
195. <i>aluminum</i>	1000 100		196 1000	
196. <i>aluminum</i>	1000 100		197 1000	
197. <i>aluminum</i>	1000 100		198 1000	
198. <i>aluminum</i>	1000 100		199 1000	
199. <i>aluminum</i>	1000 100		200 1000	
200. <i>aluminum</i>	1000 100		201 1000	
201. <i>aluminum</i>	1000 100		202 1000	
202. <i>aluminum</i>	1000 100		203 1000	
203. <i>aluminum</i>	1000 100		204 1000	
204. <i>aluminum</i>	1000 100		205 1000	
205. <i>aluminum</i>	1000 100		206 1000	
206. <i>aluminum</i>	1000 100		207 1000	
207. <i>aluminum</i>	1000 100		208 1000	
208. <i>aluminum</i>	1000 100		209 1000	
209. <i>aluminum</i>	1000 100			

Image 5



We show the length of the annotated image dataset. We show that all our images are in color (consist of 3 color channels) and we show a scatterplot of the different image sizes.

```
In [4]: print(len(image_paths))

widths = []
heights = []
channels_images = set()

for i in range(len(image_paths)):
    img = cv2.imread(image_paths[i])

    (height, width, channels) = img.shape
    channels_images.add(channels)

    widths.append(width)
```

```

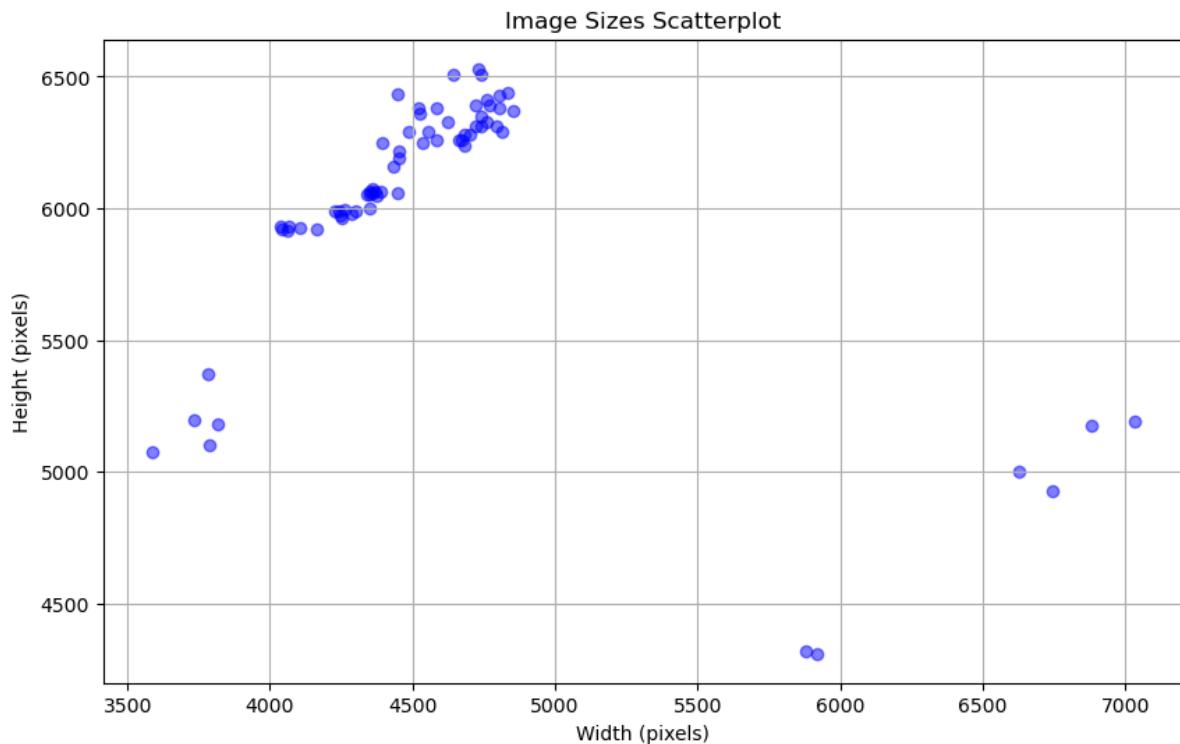
heights.append(height)

print(channels_images)

# Create a scatterplot
plt.figure(figsize=(10, 6))
plt.scatter(widths, heights, color='blue', alpha=0.5)
plt.title('Image Sizes Scatterplot')
plt.xlabel('Width (pixels)')
plt.ylabel('Height (pixels)')
plt.grid(True)
plt.show()

```

68
{3}



Analysis 2:

We need to understand our data. Since we are working with images, and more specifically handwritten text recognition, the important features of the images are stored in an XML file. The XML files contain the information about the images such as at what coordinates text regions, text lines, words etc. occur. This is important because the XML file tells us at what location on an image a certain feature may be.

First, let's show the amount of annotated words that we have in our ground truth dataset.

In [5]:

```

def get_all_xml_files(root_directory):
    xml_files = []
    for root, _, files in os.walk(root_directory):
        for file in files:
            if file.endswith(".xml"):

```

```

        xml_files.append(os.path.join(root, file))
    return xml_files

def extract_text_from_xml_gt(xml_file):
    ns = {'page': 'http://schema.primaresearch.org/PAGE/gts/pagecontent/2013-07-15'}

    # Parse the XML file
    tree = ET.parse(xml_file)
    root = tree.getroot()

    # Extract text content from all TextEquiv elements
    text_content = []
    for text_equiv in root.findall('.//page:TextEquiv', ns):
        text = text_equiv.find('page:Unicode', ns).text
        if text:
            text_content.append(text)

    return text_content

all_xml_GT = get_all_xml_files('/home/roderickmajoer/Desktop/Master/Thesis/GT')

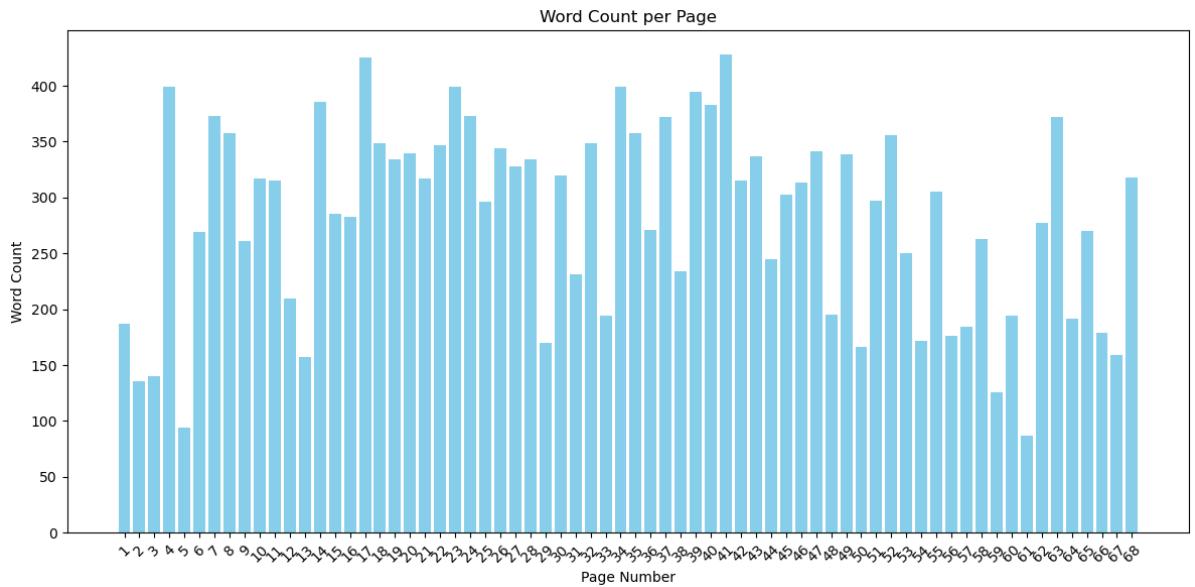
words_per_page = []
for data_gt in all_xml_GT:
    annotated_words = extract_text_from_xml_gt(data_gt)
    words_per_page.append(len(annotated_words))

pages = list(range(1, len(words_per_page) + 1))

# Create a bar plot
plt.figure(figsize=(12, 6))
plt.bar(pages, words_per_page, color='skyblue')
plt.xlabel('Page Number')
plt.ylabel('Word Count')
plt.title('Word Count per Page')
plt.xticks(pages, rotation=45)
plt.tight_layout()
plt.show()

print("Total annotated words in dataset: " + str(sum(words_per_page)))

```

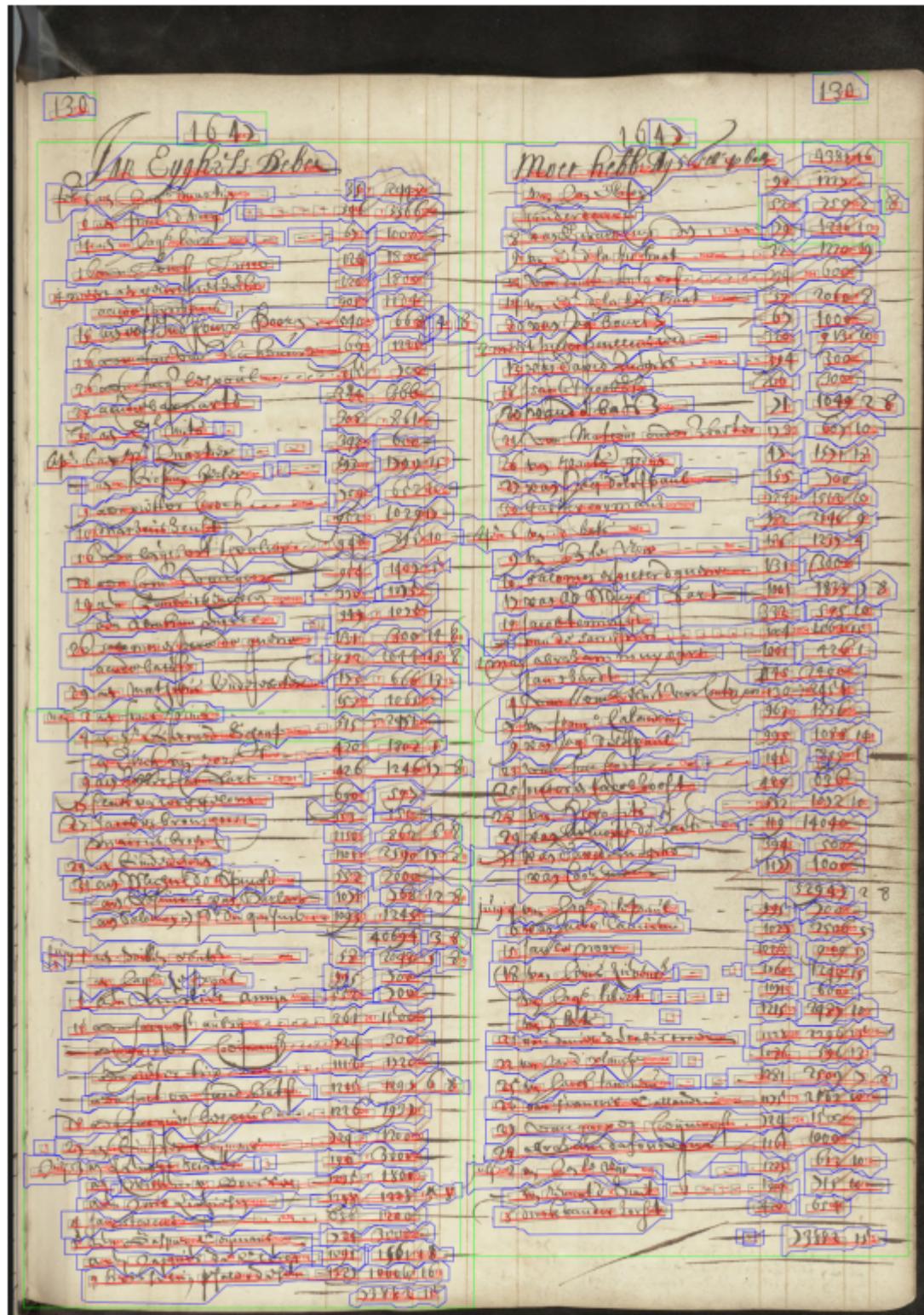


Total annotated words in dataset: 19191

To visualize these features, we plot the XML data over one of the original images so we can better understand the data. In the first cell, we do this for the ground truth XML file, which is hand annotated. In the second cell, we show this for the currently used tool to analyse these pages called Loghi. Comparing these features can give us an indication about the performance of the currently used tool.

```
In [6]: # Cell 1: Run the GT script
%run xml_on_image_GT.py
```

```
In [7]: # Cell 2: Run the Loghi script  
%run xml_on_image_loghi.py
```



```
In [8]: def extract_text_from_xml_gt(xml_file):
    ns = {'page': 'http://schema.primaresearch.org/PAGE/gts/pagecontent/2013

        # Parse the XML file
        tree = ET.parse(xml_file)
        root = tree.getroot()

        # Extract text content from all TextEquiv elements
```

```

text_content = []
for text_equiv in root.findall('.//page:TextEquiv', ns):
    text = text_equiv.find('page:Unicode', ns).text
    if text:
        text_content.append(text)

return text_content

def extract_text_from_xml_loghi(xml_file):
    ns = {'page': 'http://schema.primaresearch.org/PAGE/gts/pagecontent/2013-07-01'}

    # Parse the XML file
    tree = ET.parse(xml_file)
    root = tree.getroot()

    # Extract text content from all TextEquiv elements
    text_content = []
    for word in root.findall('.//page:Word', ns):
        text_equiv = word.find('page:TextEquiv', ns)
        if text_equiv is not None:
            text = text_equiv.find('page:Unicode', ns).text
            if text:
                text_content.append(text)

    return text_content

```

Lets now show all the words in our ground truth XML file for this image. Note that in the ground truth file, the names in the credit/debit tables are not included, so we merely focus on the money amounts and dates

```
In [9]: xml_file_path = '/home/roderickmajoer/Desktop/Master/Thesis/GT_data/55/page/all_text_gt = extract_text_from_xml_gt(xml_file_path)
print(all_text_gt)
```

```
['130', '130', '1647', 'feb 5', '35', '2448', '6', '346', '3366', '14', '6  
7', '1000', '16', '124', '1800', '4 mrt', '120', '1800', '905', '1134', '1  
5', '540', '668', '4', '8', '16', '69', '1225', '26', '155', '700', '29',  
'344', '366', '30', '308', '861', 'Ap 6', '343', '600', '593', '1799', '1  
5', '9', '709', '652', '10', '10', '952', '1029', '17', '16', '948', '895',  
'10', '18', '954', '1407', '5', '19', '330', '1075', '949', '1075', '20',  
'131', '300', '14', '8', '982', '1044', '15', '8', '29', '173', '666', '1  
3', 'may 2', '658', '1065', '4', '515', '2451', '420', '1802', '5', '9', '4  
26', '1246', '17', '8', '17', '690', '597', '27', '553', '150', '1150', '86  
2', '6', '8', '29', '1101', '2590', '17', '8', '31', '552', '2000', '1071',  
'768', '12', '8', '1093', '1245', '40694', '3', '8', 'july 1', '58', '204  
3', '5', '8', '995', '700', '5', '552', '700', '15', '261', '1500', '724',  
'300', '1112', '1720', '1215', '1291', '6', '8', '18', '1226', '1937', '2  
7', '724', '1200', 'July 3', '196', '3000', '1235', '1800', '1298', '1228',  
'5', '8', '4', '636', '1200', '5', '724', '3000', '1091', '1561', '18', '13  
27', '10006', '16', '73882', '15', '1647', '4381', '16', '92', '1227', '5  
2', '759', '7', '8', '8', '79', '1216', '10', '9', '32', '1220', '19', '1  
2', '394', '600', '14', '32', '2060', '8', '20', '67', '1000', '8 maart',  
'320', '913', '10', '13', '394', '300', '18', '260', '300', '20', '71', '10  
49', '2', '8', '21', '173', '607', '10', '26', '47', '1571', '18', '27', '1  
55', '700', '30', '724', '1513', '10', 'Ap 6', '983', '2146', '9', '9', '18  
6', '1237', '4', '10', '131', '300', '17', '1001', '1833', '7', '8', '19',  
'332', '595', '10', '1004', '1063', '15', '1 may', '1001', '426', '1', '14  
5', '2400', '4', '130', '2451', '7', '963', '1836', '9', '995', '1088', '1  
4', '23', '145', '899', '1', '25', '429', '636', '28', '572', '1072', '10',  
'29', '110', '14040', '31', '394', '500', '1177', '1000', '52947', '2',  
'8', 'juny 4', '995', '700', '8', '1027', '2506', '5', '15', '1223', '949',  
'17', '18', '1060', '1243', '15', '1191', '600', '1215', '2487', '10', '2  
1', '1228', '2296', '15', '22', '1086', '596', '13', '25', '1281', '2509',  
'7', '8', '26', '1175', '2562', '10', '27', '724', '1500', '28', '1161', '1  
000', 'july 2', '1223', '613', '10', '1304', '715', '10', '5', '420', '65  
4', '73882', '15']
```

Comparing this XML to the ground truth image above, we can see that the XML follows the tabular structure where its goes through the rows of the tables. Now we show the XML output for the Loghi tool

```
In [10]: xml_file_path = '/home/roderickmajoer/Desktop/Master/Thesis/loghi/data/55/pa  
all_text_loghi = extract_text_from_xml_loghi(xml_file_path)  
print(all_text_loghi)
```

['130', 'Jan', 'Eijghels', 'Deben', 'feb:', '5', 'aen', 'Page.', 'Martijn', '35', 'f', '2448', '---', '---', '---', '---', '346', 'f', '3366', '6', 'aen', 'Franc.', "d'vicq", '-67', '1000', '---', '---', '14', 'aen', 'Jage.', 'bours', '1800', '124', '160n', 'Lourisz', 'Lusse', '1800', '1120', 't', 'meert', 'aen', 'geurt', 'Pietersz', '905', '1134', 'aende', 'Coupans', '668', '8', '4', '15', 'aende', 'oost', "Ind'", 'Compe.', 'Hoorn', '540', '1225', '16', 'aent', 'fait', 'van', 'Wickenoort', '69', '26', 'aot', 'Jacq', 'lespoul', '55', 'f', '100', '844', '366', '29', 'aende', 'barnarts', '308', 'f', '861', '---', 'mits', '30', 'aen', '600', '---', '342', '---', 'Quartier', 'apn.', 'Caextpr.', '179', '9', 'f', '15', '692', '---', '---', '---', 'aen.', 'Pios uim', 'Heller', '652/0', 'p09', '9', 'aen', 'pietter', 'Cloeck', 'f', '52', '10291', '10', 'Matheus', 'Senst', 'Ro10', '16', 'aer', 'hijsbert', 'theulinx', 'f', '948', '954', '1407', 'f', '5', '28', 'aen', 'Corn.', 'vangoor', '1075', '330', '19', 'aen', 'Comboit', 's', 'Jacobsen', '949', 'f', '1078', '*', '---', '---', 'aen', 'Abraham', 'engels', '131', '300', 'f', '148', '26', 'salo', 'mos', 'es', 'perco', 'de', 'quena', '+', '982', '1044f158', 'aende', 'baths', '173', 'f', '666', '13', '29', 'aen', 'mathasus', 'Ondesebote r', '658', '1065', 'maiij', '2', 'aen', 'Frans', 'sprins', '4', 'an.', 'h r.', 'Gsercard:', 'Schaap.', '---', '---', '515.', '2451', 'ean', 'Dirck', 'van', 'Horst', '---', '420', 'f', '1802', '5', '426', '124613', '8', '9', 'aend', 'robber', 'Caffart', '690', '597', 'Een', 'Claes', 'van', 'woggelom', '150', '553', '27', 'Jacobvn', 'bronchorst', '1186', '862', '68', 'Marcus', 'broes', '1101', '2590', '17', '8', '29', 'aen', 'Ginderdeuren', '552', '2000', '31', 'aen', 'Michiel', 'de', 'Spinose', '768', 'f', '128', '1071', 'aen', 'Josumus', 'van', 'Barlaer', 'aen', 'Salomon', 'en', 'Po.', 'du', 'quesmo', '1093', 'f', '1245', '40694', '38', 'Junij', 'aen', 'Willem', 'wats', '58', '2042', '5', '58', '5.', '295', '300', 'xgut', 'of', 'aen', 'hays.', '5552', '100', '5', 'aen', 'michiel', 'Amijn', '15', 'adnjacquesz', 'aubrij.', '---', '---', '---', '261', '1500', 'adagaiper', 'Coeijmansz.', '124', '---', '300-', 'adrpieter', 'tripdn', 'f', '1112', '1720', '1215', '129', '&', '6', '8.', 'aen', 'port', 'van', 'fand', 'Bats.', '18', 'adnfacque', 'Gespoul', '---', '---', '1226', '1931', '27', 'aen', 'Gapper', 'Coijmans', 'f24', 'f', '1200', '1', '196', 'f', '3000', '---', 'Juij:', 'van', 'Hena rt', 'Slischer', 'a', 'Willem', 'v:', 'Borssel', '1235', 'f', '1800', '1298', '1228', 'f', '55', 'aen', 'Joris', 'Linniesen', '---', '636', '1200', '4', 'Jan', 'stevens', '724', '3000-', '5', 'aen', 'Gaspur', 'Ceijmans', '156:l18', '191', 'aen', 'Jacques', 'de', 'Clercq', '9', 'hem', 'selven', 'pesalvo', 'desen', '1527', '10006', 'f', '16', '1785½', '15', 'Moet', 'hebb:', 'By5', 'Feb:', '10', 'bal', 'van', 'Jan', 'Elise', 'ginderdeurg', '8', '8van', 'Gerard', 'Luysten', '9', 'van', 'Do:', 'd'le', 'histraat', '---', '32', '1220-19', 'th', 'van', 'danit', 'Antgers.', '---', '---', '---', '---', '394', '600', '32f', '2060,,8', '14', 'van.', 'D:', 'dla', 'bistraat', '67', '1000', '20', 'van', 'Jaqe.', 'boursz', 'P20', '9', '13', '10', 'meert', 'Pieter', 'Poullens', 'wed', '394', '300', '13', 'van', 'David', 'Zuegers', '300', '266', '18', 'Isack', 'Jacobssz', '1049', '28', '21', '20', 'vande', 'bats', '60', '7', '10', '21', 'van', 'Mathout', 'onder', 'Waster', '173', '47', '1571', '18', '26', 'van', 'ppauls', 'godijn', '155', '100', '27', 'van', 'ssaij', 'delespaul', 'f', '24', '1513', '9', '30', 'gasster', 'coyman s', '182', '2146', '9', 'Apo.', '6', 'vn', 'd', 'bats', '186', '1237', '4', '9', 'van', 'Iole', 'Boi', 'f', '100', 'to', 'Salomes', '&', 'pieterdquene', '131', '18', '1001', '18337', 'dart', '17', 'van', 'ad', 'Muys', '595', '10', '932', '19', 'Jacob', 'tenmin', 'van', 'de', 'sarresstiers.', '---', '---', '---', '1004', '1663', '15', '1', '1001', '4261', '(may', 'abraham', 'muygart', 'R45', '2400', 'Jan', 'Bardt', '4', 'vanAem', 'Selust', 'voor', 'Conten', 'por.', '130', '2451', '963', '1836', 'van', 'Franc o.', 'Calandrinx', '995', '1088', '14', '9', 'van', '10g', 'daelefpaul', '8

```
991', '145', '--', 'f', '23', 'van', 'Jan', 'Chart', '636', '429', 'V', 'poc  
ctor/', 'en', 'Japob', 'hooft', 'f512.', '1012', '10', '28', 'van', 'Vugo',  
'fits', '119', '14040', '29', 'pag', 'Aernoits', 'de', 'Zaet', '294', '50  
0', '3lse', 'an', 'David', 'Lntgers', '1000', 'pi3)', 'vande', 'Coeiman',  
'5294', 'junij', '4', 'van', 'Jage.', "d'le", 'paul', '395', '700', '2506  
5', 'van', 'Jacob', 'Tamieau', '1027', '1223', '949', '17', '15', 'Janle',  
'noor', '1060', '1242', '15', '1228', 'I8', 'van', 'Joris', 'Lievensz',  
'1191', '600', '1228', '229615-', '21', 'van', 'daniel', 'de',  
'labistraet', '1086', '596', '13', '22', 'pan', 'Jan', "d'", 'planch  
e', '1281', '2509', '7', '8', '25', 'van', 'Jacob', 'tamuneau',  
'26', 'van', 'Francois', 'Callandri', '1175', '256210', '27', 'vangasper',  
'Coeijmunss', '724', '150', '28', 'abraham', 'dafondeau', '116', '1000', '6  
12', '10-', '1223', 'Julij', '2', 'van', 'Jan', 'le', 'Noij', '--', '71510-  
', '-1', '1304.', 'van', 'remont', "d'", 'Smit', '420', '65  
4', '5', 'dirck', 'vander', 'horst', '17882', '15', '438116', '9  
2.', '1227-', '52', '759r.', '1216', '10', '79', '47', '1647', '130']
```

Looking at the Loghi XML output and image, we can see that (besides the recognized names which we discarded in the ground truth) a lot of false positives are detected, mainly '!' and '-' characters that are typically detected due to small marks/stripes on the image. In order to get better results, we should try to pre-process our image such that these marks/stripes are not detected as text.

For now let's just get rid of the elements in the XML files that don't contain numbers. This should give us a better comparison of the numbers found and might tell us something about the order of the Loghi XML file.

```
In [11]: filtered_list = [item for item in all_text_loghi if any(char.isdigit() for c  
print(filtered_list)
```

```
[ '130', '5', '35', '2448', '346', '3366', '6', '-67', '1000', '14', '1800',
'124', '160n', '1800', '1120', '905', '1134', '668', '8', '4', '15', '540',
'1225', '16', '69', '26', ',55', '100', '844', '366', '29', '308', '861',
'30', '600', '342', '179', '9', '15', '692', '652/0', 'p09', '9', '52', '10
291', '10', 'Ro10', '16', '948', '954', '1407', '5', '28', '1075', '330',
'19', '949', '1078', '131', '300', '148', '26', '982', '1044f158', '173',
'666', '13', '29', '658', '1065', '2', '4', '515.', '2451', '420', '1802',
'5', '426', '124613', '8', '9', '690', '597', '150', 'S53', '27', '1186',
'862', '68', '1101', '2590', '17', '8', '29', '552', '2000', '31', '768',
'128', '1071', '1093', '1245', '40694', '38', '58', '2042', '5', '58',
'5.', '295', '300', '5552', '100', '5', '15', '261', '1500', '124', '300-',
'1112', '1720', '1215', '129', '6', '8.', '18', '1226', '1931', '27', 'f2
4', '1200', '1', '196', '3000', '1235', '1800', '1298', '1228', '55', '63
6', '1200', '4', '724', '3000-', '5', '156:l18', '191', '9', '1527', '1000
6', '16', '1785½', '15', 'By5', '10', '8', '8van', '9', '32', '1220-19', '3
94', '600', '32f', '2060,,8', '14', '67', '1000', '20', 'P20', '9', '13', '1
0', '394', '300', '13', '300', '266', '18', '1049', '28', '21', '20', '60',
'7', '10', '21', '173', '47', '1571', '18', '26', '155', '100', '27', '24',
'1513', '9', '30', '182', '2146', '9', '6', '186', '1237', '4', '9', '100',
'131', '18', '1001', '18337', '17', '595', '10', '932', '19', '1004', '166
3', '15', '1', '1001', '4261', 'R45', '2400', '4', '130', '2451', '963', '1
836', '995', '1088', '14', '9', '10g', '8991', '145', '23', '636', '429',
'f512.', '1012', '10', '28', '119', '14040', '29', '294', '500', '31se', '1
000', 'pi3)', '5294', '4', '395', '700', '25065', '1027', '1223', '949', '1
7', '15', '1060', '1242', '15', 'I8', '1191', '600', '2487', '10', '1215',
'1228', '229615-', '21', '1086,,', '596', '13', '22', '1281', '2509', '7',
'8', '25', '26', '1175', '256210', '27', '724', '150', '28', '116', '1000',
'612', '10-', '1223', '2', '71510-', '-1', '1304.', '420', '654', '5', '178
82', '15', '438116', '92.', '1227-', '52', '759r.', '1216', '10', '79', '4
7', '1647', '130']
```

```
In [12]: print(len(all_text_gt))
print(len(filtered_list))
```

```
341
314
```

Now, The list of numbers in the Loghi detected text is shorter than the ground truth list. This is because some of the numbers in the ground truth are actually not recognized as numbers by Loghi. This is another error in Loghi that we should try to improve. Other things we notice are that there are some words/numbers that become merged in Loghi's detection. The last thing that we can see from the images and lists, is that Loghi is not able to capture the table structure of the page correctly thus creating a wrong order of words in the list.

Some of the things we noticed going wrong in Loghi's output:

1. False positives detected (typically due to marks / stripes on the image that are recognized as text)
2. Numbers are not detected as numbers but rather as text / other characters
3. Words/numbers are merged or seperated (i.e. 16 becomes 1, 6 or 2, 8 becomes 28 etc.)
4. Numbers not correctly recognized (i.e. 16 becomes 18 etc.)
5. Output not in the correct layout

Analysis 3:

We can try to find the Word Error Rate (WER) and Character Error Rate (CER) from the output of Loghi. However, since we have a shorter list as Loghi output than the ground truth and we know the order is not correct, we can expect a very bad score.

```
In [13]: wer = jiwer.wer(all_text_gt[:314], filtered_list)
cer = jiwer.cer(all_text_gt[:314], filtered_list)
print("Word Error Rate:", wer)
print("Character Error Rate:", cer)
```

```
Word Error Rate: 0.9783950617283951
Character Error Rate: 1.0684150513112884
```

We will check if there are matching words in both lists. Here we do not look at word order, merely if a word occurs somewhere in both lists. This at least gives us an idea whether Loghi has some matching words with the ground truth.

```
In [14]: def count_matching_words(list1, list2):
    # Convert lists to dictionaries to count occurrences
    dict1 = {}
    dict2 = {}

    # Count occurrences in list 1
    for word in list1:
        dict1[word] = dict1.get(word, 0) + 1

    # Count occurrences in list 2
    for word in list2:
        dict2[word] = dict2.get(word, 0) + 1

    # Find the intersection of the two dictionaries
    matching_words = set(dict1.keys()).intersection(set(dict2.keys()))

    # Count the occurrences based on the minimum count in both lists
    total_count = sum(min(dict1.get(word, 0), dict2.get(word, 0)) for word in matching_words)

    return total_count
```

```
In [15]: matching_count = count_matching_words(all_text_gt, filtered_list)
print("Number of matching words:", matching_count)
```

```
Number of matching words: 215
```

We find 215 matching words for this page, out of 314 total words in the (filtered) Loghi output. So about 68% of Loghi's output is also found in the ground truth. However, we still have no idea about the correct word order.

We can also calculate the jaccard similarity score

```
In [16]: def jaccard_similarity(list1, list2):
    set1 = set(list1)
    set2 = set(list2)
    intersection = len(set1.intersection(set2))
    union = len(set1) + len(set2) - intersection
    return intersection / union if union != 0 else 0
```

```
In [17]: print("Jaccard Similarity:", jaccard_similarity(all_text_gt, filtered_list))
Jaccard Similarity: 0.4708029197080292
```

Now lets show the mean metrics for all images

```
In [18]: all_xml_GT = get_all_xml_files('/home/roderickmajoer/Desktop/Master/Thesis/GT')
all_xml_loghi = get_all_xml_files('/home/roderickmajoer/Desktop/Master/Thesis/Loghi')

list1_filenames = [os.path.basename(path) for path in all_xml_GT]
list2_filenames = [os.path.basename(path) for path in all_xml_loghi]
if list1_filenames == list2_filenames:
    print("The lists are in the same order based on filenames.")
```

The lists are in the same order based on filenames.

```
In [19]: def compute_mean_metrics(all_xml_GT, all_xml_loghi):

    wer_total = 0
    cer_total = 0
    matching_count_total = 0
    jaccard_similarity_total = 0
    num_images = 0

    for data_gt, data_loghi in zip(all_xml_GT, all_xml_loghi):
        all_text_gt = extract_text_from_xml_gt(data_gt)
        all_text_loghi = extract_text_from_xml_loghi(data_loghi)
        filtered_list = [item for item in all_text_loghi if any(char.isdigit for char in item)]
        wer_total += jiwer.wer(all_text_gt[:len(filtered_list)], filtered_list)
        cer_total += jiwer.cer(all_text_gt[:len(filtered_list)], filtered_list)
        matching_count_total += count_matching_words(all_text_gt, filtered_list)
        jaccard_similarity_total += jaccard_similarity(all_text_gt, filtered_list)
        num_images += 1

    mean_wer = wer_total / num_images
    mean_cer = cer_total / num_images
    mean_matching_count = matching_count_total / num_images
    mean_jaccard_similarity = jaccard_similarity_total / num_images

    return mean_wer, mean_cer, mean_matching_count, mean_jaccard_similarity
```

```
In [20]: mean_wer, mean_cer, mean_matching_count, mean_jaccard_similarity = compute_mean_metrics(all_xml_GT, all_xml_loghi)

print("Mean Word Error Rate:", mean_wer)
print("Mean Character Error Rate:", mean_cer)
print("Mean Matching Count:", mean_matching_count)
print("Mean Jaccard Similarity:", mean_jaccard_similarity)
```

```
Mean Word Error Rate: 0.9810198160684107
Mean Character Error Rate: 1.0649680926496443
Mean Matching Count: 0.6609155616036941
Mean Jaccard Similarity: 0.4582747425528701
```

As we expected, we can see a bad score for WER and CER (mainly because of the word order) and we see that the mean matching count and jaccard similarity show that loghi does manage to get some output correct.

Analysis 4:

Now we will match the bounding boxes of the table cells in the ground truth with the bounding boxes of words found by loghi based on their intersection over union (IoU) value. The IoU value is a metric that can be used to determine the overlap between two sets (in this case, between two bounding boxes). This way, we can see the correct order. We will do this for the first image again.

```
In [21]: data_gt, data_loghi = main()
```

```
In [22]: print(data_gt)
print(data_loghi)
```

['130', '130', '1647', '1647', 'feb 5', '35', '2448', '6', '346', '3366', '14', '67', '1000', '16', '124', '1800', '4 mrt', '120', '1800', '905', '1134', '15', '540', '668', '4', '8', '16', '69', '1225', '26', '155', '700', '29', '344', '366', '30', '308', '861', 'Ap 6', '343', '600', '593', '1799', '15', '9', '709', '652', '10', '952', '1029', '16', '948', '895', '18', '954', '1407', '5', '19', '330', '1075', '949', '1075', '20', '131', '300', '14', '982', '1044', '29', '173', '666', '13', 'may 2', '658', '1065', '4', '515', '2451', '420', '1802', '5', '9', '426', '1246', '8', '17', '690', '597', '27', '553', '150', '1150', '862', '6', '29', '1101', '2590', '17', '8', '31', '552', '2000', '1071', '768', '12', '1093', '1245', '40694', '3', 'july 1', '58', '2043', '5', '8', '995', '700', '5', '552', '700', '1', '261', '1500', '724', '300', '1112', '1720', '1215', '1291', '6', '8', '18', '1226', '1937', '27', '724', '1200', 'July 3', '196', '3000', '1235', '1800', '1298', '1228', '5', '4', '636', '5', '724', '3000', '1091', '1561', '1327', '10006', '16', '15', '4381', '92', '1227', '52', '759', '8', '8', '79', '1216', '10', '9', '32', '1220', '12', '394', '600', '14', '32', '2060', '20', '67', '1000', '8 maart', '320', '913', '10', '13', '394', '300', '18', '260', '300', '20', '71', '1049', '8', '21', '173', '607', '10', '26', '47', '1571', '18', '27', '155', '700', '30', '724', '1513', '10', 'A p 6', '983', '2146', '9', '9', '186', '1237', '4', '10', '131', '300', '1', '7', '1001', '1833', '7', '19', '332', '595', '10', '1004', '1063', '15', '1 may', '1001', '426', '145', '2400', '4', '130', '2451', '7', '963', '1836', '9', '995', '1088', '14', '23', '145', '899', '25', '429', '636', '28', '572', '1072', '10', '29', '110', '14040', '31', '394', '500', '1177', '1000', 'juny 4', '995', '700', '8', '1027', '2506', '15', '1223', '949', '17', '1', '8', '1060', '1243', '15', '1191', '600', '1215', '2487', '10', '21', '1228', '2296', '22', '1086', '596', '13', '25', '1281', '2509', '7', '8', '2', '6', '1175', '2562', '27', '724', '1500', '28', '1161', '1000', 'july 2', '1223', '613', '10', '1304', '715', '5', '420', '654', '15']
['130', '130', '1647', '47', 'feb:', 'f', '2448', '6', '346', '3366', '14', '-67', '1000', '160n', '124', '1800', 'meert', '1120', '1800', '905', '1134', '15', '540', '668', '4', '8', '16', '69', '1225', '26', '55', '100', '29', '844', '366', '30', '308', '861', 'apn.', '342', '600', '692', '179', '15', '9', 'p09', '652/0', '10', '52', '10291', '16', '948', 'Ro10', '28', '954', '1407', '5', '19', '330', '1075', '949', '1078', '26', '131', '300', '148', '982', '1044f158', '29', '173', '666', '13', 'maijs', '658', '1065', '4', '2451', '420', '1802', '5', '9', '426', '124613', '8', 'Een', '690', '597', '27', '553', '150', '1186', '862', '68', '29', '1101', '2590', '17', '8', '31', '552', '2000', '1071', '768', '128', '1093', '1245', '40694', '4', '38', 'Junij', '58', '2042', '5', '58', '295', '300', '5', '552', '100', '15', '261', '1500', '124', '300', '1112', '1720', '1215', '129', '6', '8.', '18', '1226', '1931', '27', 'f24', '1200', 'Juij:', '196', '3000', '1', '235', '1800', '1298', '1228', '55', '4', '1200', '5', '724', '3000', '191', '156:118', '1527', '10006', '16', '15', '438116', '92.', '1227', '52', '759r.', '8', '8van', '79', '1216', '10', '9', '32', '1220-19', 'th', '394', '14', '32f', '2060,,8', '20', '67', '1000', 'meert', 'P20', '13', '1', '13', '394', '300', '18', '266', '300', '20', '21', '1049', '28', '21', '173', '60', '10', '26', '47', '1571', '18', '27', '155', '100', '30', '2', '4', '1513', '9', 'Apo.', '182', '2146', '9', '9', '186', '1237', '4', 'to', '131', '100', '17', '1001', '18337', '18', '19', '932', '595', '10', '1004', '1663', '15', '(may', '1001', '4261', 'R45', '2400', '4', 'por.', '2451', 'van', '963', '1836', '9', '995', '1088', '14', '23', '145', '8991', 'V', '429', '636', '28', 'f512.', '1012', '10', '29', '119', '14040', '31s e', '294', '500', 'pi3)', '1000', 'junij', '395', '700', 'van', '1027', '25065', '15', '1223', '949', '17', 'I8', '1060', '1242', '15', '1191', '600', '1215', '2487', '10', '21', '1228', '229615-', '22', '1086,,', '596', '13']

```
'25', '1281', '2509', '7', '8', '26', '1175', '256210', '27', '724', '150',
'28', '116', '1000', 'Julij', '1223', '612', '10-', '1304.', '71510-', '5',
'420', '654', '15']
```

Now we will perform the same tests as before

```
In [23]: wer = jiwer.wer(data_gt, data_loghi)
cer = jiwer.cer(data_gt, data_loghi)
matching_count = count_matching_words(data_gt, data_loghi) / len(data_loghi)
jaccard_similarity = jaccard_similarity(data_gt, data_loghi)

print("Word Error Rate:", wer)
print("Character Error Rate:", cer)
print("Matching Count:", matching_count)
print("Jaccard Similarity:", jaccard_similarity)
```

```
Word Error Rate: 0.38198757763975155
Character Error Rate: 0.2139689578713969
Matching Count: 0.6495176848874598
Jaccard Similarity: 0.45
```

Now we have way better scores for word error and character error. However we could not match every ground truth word to a loghi word based on IoU. Only 311 out of 341 ground truth regions were matched to a loghi region. But this does show that if we can find the correct layout from the loghi output, the performance will drastically increase.

```
In [ ]:
```