

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. 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

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/roderickmajoor/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/roderickmajoor/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

		Creditor's name & balance			
Debt	Credit	Debt	Credit	Balance	
10 327	Lambert De Grootveld	329	916	6 acnt	
12 329	Jan. fidei. Coop.	—	20	10 acnt	
2 325	Abt. visscher Grootveld Grootveld	327	312	7 acnt	
11 326	Jan. fidei. Coop. Grootveld	326	404	6 acnt	
10 327	Mark Peeters Grootveld	327	181	6 account	
11 329	Pieter van Zeeft	—	227	10 acnt	
2 325	Wouter Grootveld	325	228	17 8 account	
17 328	Ant. Coop. de Grootveld	328	26	13 9 old	
9 328	Fran. Grootveld	328	1060	9 acnt	
28 329	Wouter Grootveld	329	1060	9 acnt	
17 329	Abd. Grootveld	—	1060	9 acnt	
11 326	G. Grootveld	326	27	6 account	
10 328	Bergenius Hendrik	—	8	6 acnt	
2 324	Jan. Grootveld en C.	324	—	1 6	
10 329	Jan. Grootveld	—	—	—	
11 327	Gerardus Grootveld	327	217	12 account	
11 327	Gerardus Grootveld	327	2014	12 account	
11 323	Abd. Grootveld	323	422	11 9 old	
4 326	Wouter Grootveld	326	412	16 8 acnt	
2 326	Jacob Grootveld	326	21	12 8 acnt	
<hr/>					
6	Grootveld Jacobus				
1 324	Grootveld B. v.d.	324	14891	9 acnt	
3 321	B. Grootveld	320	14891	9 acnt	
9 325	Josph. Grootveld de Grootveld	325	21	16 8 account	
10 326	Josph. Grootveld	326	194	15 8 acnt	
16 326	Josph. Grootveld	326	1	8 acnt	
10 327	Josph. Grootveld	327	20	11 8 account	
10 327	Ab. Grootveld	327	788	11 account	
10 328	Abd. Grootveld	328	807	11 acnt	
2 328	Grootveld Jacobus	328	357	3 8 account	
2 327	Grootveld Jacobus	327	351	8 acnt	
2 327	Grootveld Jacobus	327	1206	11 account	
22 329	Abd. Grootveld	329	156	4 account	
22 329	Jan. v. Grootveld v. Grootveld	329	156	4 account	
2 327	Jacob Grootveld	327	1317	11	
20 328	Josph. Van Grootveld	328	65	8 account	
7 327	Jacob Grootveld	327	160	7 acnt	
22 328	Josph. Grootveld v. Grootveld	328	1766	8 acnt	
10 326	Abd. Grootveld v. Grootveld	326	2	4 acnt	
11 329	Jan. v. Grootveld	329	—	— account	
2 327	Jacob Grootveld	327	4	10 acnt	
22 326	Jacob Grootveld v. Grootveld	326	2106	11 acnt	
<hr/>					
10 320					

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 Provintial Vast Kasse 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	300	Muzzen	2000,-		
5 - H	120	Dito	-		
6 - I	120	Dito	-		
6 - K	120	Dito	-		
7 - L	120	Dito	-		
8 - M	60	Verlorenen	1200,-		
8 - N	600	Nieuwe Verlorenen	480,-		
9 - O	120	Muzzen	1200,-		
11 - P	500	Amsterdamsche Duraties	1000,-		
12 - AA	300	Muzzen	1200,-		
15 - BB	300	Dito	-		
16 - DD	300	Dito	-		
17 - EE	300	Dito	-		
18 - FF	300	Dito	-		
19 - GG	300	Dito	-		
18 - HH	300	Dito	-		
19 - JJ	300	Dito	-		
19 - KK	300	Dito	-		
20 - LL	300	Dito	-		
20 - MM	300	Dito	-		
21 - NN	300	Dito	-		
22 - PP	300	Dito	-		
23 - QQ	300	Dito	-		
24 - RR	300	Dito	-		
24 - SS	2400	Belassen Kapel 11	6280.000		
32 - TT	1137	Nieuwe Fransche Crone Kapel 12	2501.400		
			22 568.914 £		

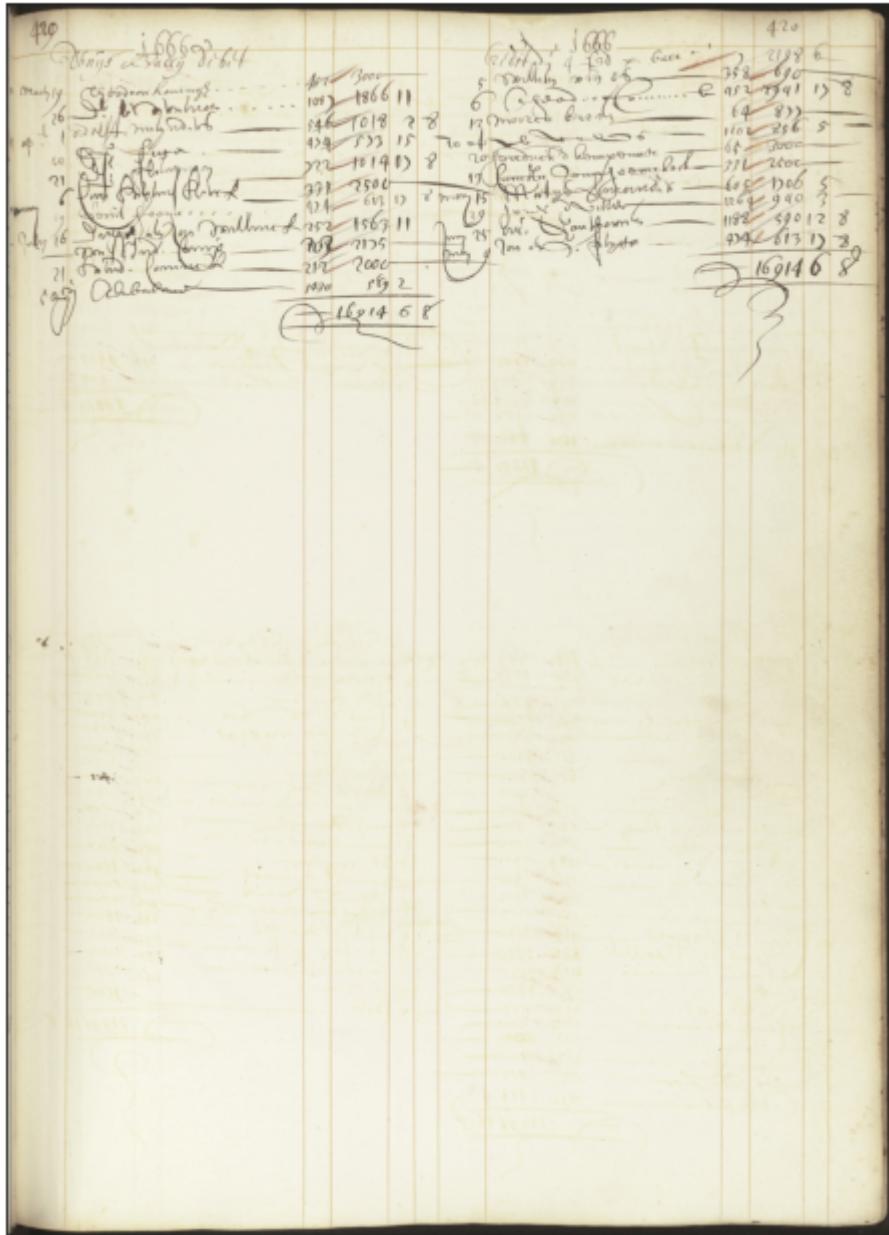
Image 3

AenWyzing der Winsten op 28 Januarij 1763		AenWyzing der Winsten op 28 Januarij 1763	
fol. 1 tot 3		fol. 1 tot 3	
39	2	op de Nieuwe Crone	374,-
41	3	Goud-Ducaten a 500 fl.	1224.11,-
43	4	op de Goud-Ducaten	1,-
46	5	Goud-Ducaten a 1000 fl.	623.50,-
48	6	op de Goud-Ducaten a 1000 fl.	1176.13,-
51	7	op de Goud-Ducaten a 1000 fl.	2263.12,-
56	8	Goud-Ducaten	404.50,-
64	11	De Goud-Crone	1033,-
78	13	De Muzzen	4396,-
86	14	Gaste-Dreigolden	105.15,-
28	15	Spes-Crone	17.63.10,-
95	16	Verlorenen	11,-
99	18	Goud-Ducaten-Restanten	143.6,-
100	17	Goud-Ducaten Restanten	162.6,-
102	17	Goud-Eurogen	176.10,-
107	17	Goud-Crader	2591.12,-
			181757.61
			2663.4,-
		Van de Goud-Ducaten restanten van 240.000 fl. 241.300 Salder mit Augen 123	14357.14,-
		Van kastervandelft en Grondes voor Kosten Voor Stokken Geld van Jan 1762	126,-
			570,-
			26535.4,-
		Aff voldoet 100,- voor pma 100,- 2 obligaties op goud-Ducaten-Restanten	2351.6,-
		en 100,- op de Nieuwe Crone-Restanten Vergoedt 124.16,-	124.16,-
		af Aug 8.9.17 124.16,- op 28.12.7.8 / 2773.15,-	2773.15,-
			Tijverdienst 124.16,-

Image 4

	620	166		620
1. <i>François Desirée leottt</i>			6/1/18 192 19 8	
2. <i>Proprietary aluminum</i>	61 140		26 1005 78	
3. <i>Stainless steel</i>	120 725		27 1000 100	
4. <i>Aluminum frame glass</i>	450 125		28 1000 100	
5. <i>Aluminum</i>	860 120		29 1000 100	
6. <i>Aluminum</i>	117 100		30 1000 100	
7. <i>Metal frame</i>	200 1000		31 1000 100	
8. <i>Aluminum</i>	934 600		32 1000 100	
9. <i>Aluminum</i>	167 350		33 1000 100	
10. <i>Cat</i>	102 2010 18		34 1000 100	
11. <i>Orchard fence</i>	743 1110		35 1000 100	
12. <i>Aluminum</i>	360		36 1000 100	
13. <i>Aluminum</i>	1422 1300		37 1000 100	
14. <i>Aluminum</i>	831 1319	5	38 1000 100	
15. <i>Aluminum</i>	675 1170		39 1000 100	
16. <i>Aluminum</i>	878 500		40 1000 100	
17. <i>Aluminum</i>	113 585	10	41 1000 100	
18. <i>Aluminum</i>	332 1440		42 1000 100	
19. <i>Aluminum</i>	112 500		43 1000 100	
20. <i>Aluminum</i>	912 1800		44 1000 100	
21. <i>Aluminum</i>	12653 3		45 1000 100	
22. <i>Aluminum</i>	916 268	5	46 1000 100	
23. <i>Aluminum</i>	118 2113	3	47 1000 100	
24. <i>Aluminum</i>	127 1410		48 1000 100	
25. <i>Aluminum</i>	109 2000		49 1000 100	
26. <i>Aluminum</i>	102 623	3	50 1000 100	
27. <i>Aluminum</i>	172 300		51 1000 100	
28. <i>Aluminum</i>	181 1219	1	52 1000 100	
29. <i>Aluminum</i>	103		53 1000 100	
30. <i>Aluminum</i>	121 1015		54 1000 100	
31. <i>Aluminum</i>	124 1015		55 1000 100	
32. <i>Aluminum</i>	934 1150		56 1000 100	
33. <i>Aluminum</i>	226 1010		57 1000 100	
34. <i>Aluminum</i>	120 630		58 1000 100	
35. <i>Aluminum</i>	128 300		59 1000 100	
36. <i>Aluminum</i>	105 1507	10	60 1000 100	
37. <i>Aluminum</i>	1270 1107	16	61 1000 100	
38. <i>Aluminum</i>	65 525	12	62 1000 100	
39. <i>Aluminum</i>	115 725		63 1000 100	
40. <i>Aluminum</i>	206 790		64 1000 100	
41. <i>Aluminum</i>	1218 601	10	65 1000 100	
42. <i>Aluminum</i>	162 2900		66 1000 100	
43. <i>Aluminum</i>	164 2912	8	67 1000 100	
44. <i>Aluminum</i>	164 2912	8	68 1000 100	
			40732 10	

Image 5



We show the length of the image dataset, print the unique sizes of the images in it and show the color channels to show if all images are in color.

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

unique_sizes = set()
channels_images = set()

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

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

```
print(unique_sizes)
print(channels_images)

68
{(6440, 4832), (5990, 4241), (6159, 4431), (6370, 4852), (6380, 4802), (6310, 4722), (5923, 4041), (6434, 4446), (6046, 4374), (6310, 4792), (6380, 4582), (6002, 4349), (6219, 4451), (6280, 4702), (5927, 4105), (6260, 4672), (6390, 4772), (5370, 3782), (6250, 4536), (6350, 4742), (4318, 5882), (6430, 4802), (6056, 4359), (5178, 6882), (5931, 4035), (6260, 4586), (6061, 4349), (6053, 4351), (6290, 4486), (5929, 4065), (6260, 4662), (6290, 4556), (6061, 4367), (6075, 4360), (6510, 4642), (5189, 7031), (6310, 4742), (5993, 4263), (5992, 4228), (5963, 4250), (6191, 4454), (6290, 4812), (6360, 4526), (5980, 4287), (6390, 4722), (4927, 6747), (5001, 6629), (5074, 3587), (6510, 4742), (6380, 4520), (6280, 4682), (6052, 4342), (5923, 4164), (5971, 4247), (6410, 4762), (6530, 4732), (6247, 4392), (6063, 4391), (6330, 4622), (4310, 5921), (5100, 3785), (5179, 3816), (6059, 4445), (5991, 4301), (6240, 4682), (5918, 4061), (6330, 4762), (5196, 3734)}
{3}
```

Analysis 2:

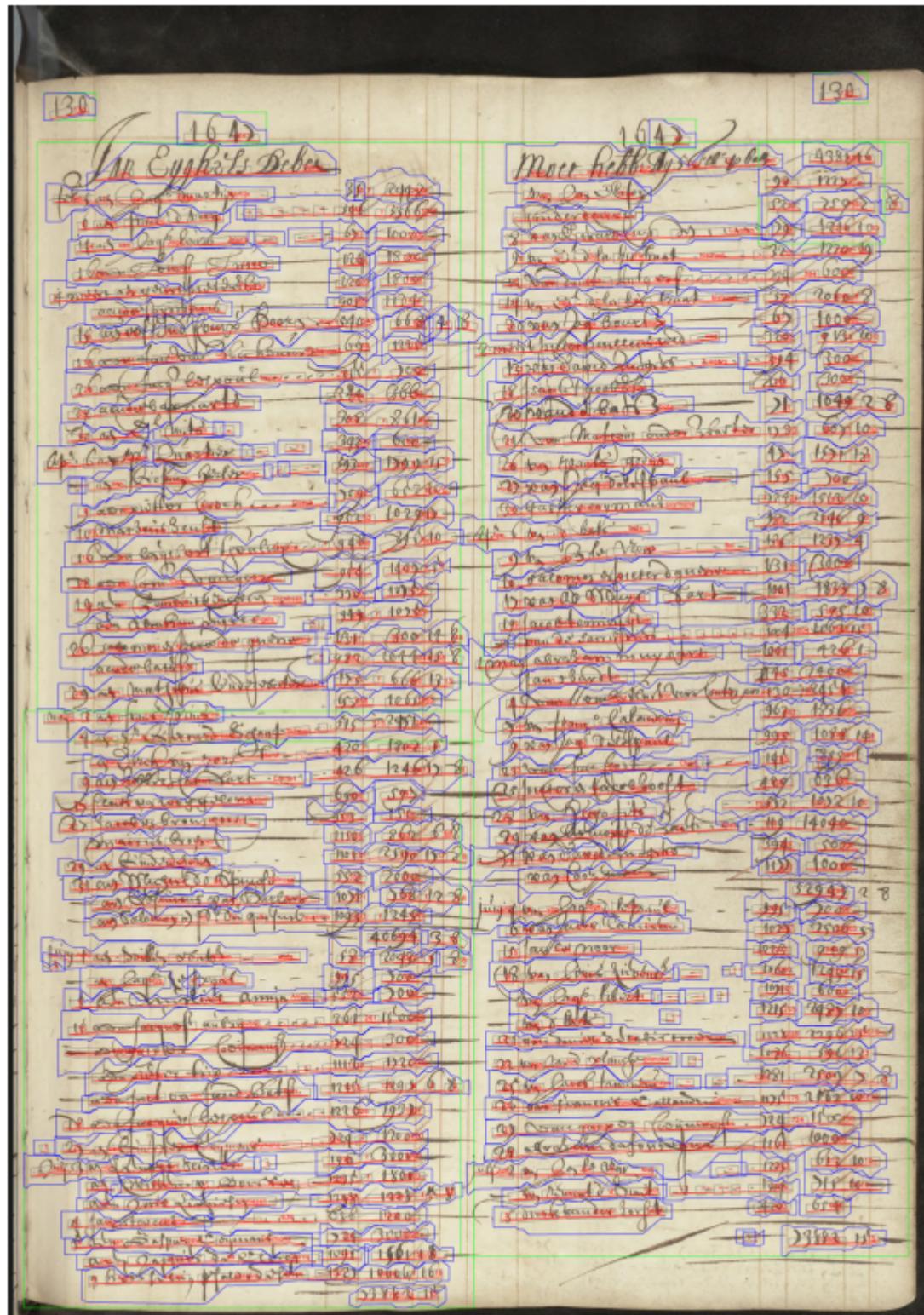
We need to understand our data. Since we are working with images, the important features of the images are stored in an XML file. The XML files contain the information about the images such as at what coordinates text regions, text lines, words etc. occur. To visualize these features, we plot the XML data over the original image 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 Loghi. Comparing these features can give us an indication about the performance of the currently used tool.

```
In [5]: # Cell 1: Run the GT script
%run xml_on_image_GT.py
```

		164
1.	1. <u>Common Name</u>	30
2.	2. <u>Family</u>	20
3.	3. <u>Genus</u>	20
4.	4. <u>Species</u>	20
5.	5. <u>Common Name</u>	10
6.	6. <u>Family</u>	10
7.	7. <u>Genus</u>	10
8.	8. <u>Species</u>	10
9.	9. <u>Common Name</u>	10
10.	10. <u>Family</u>	10
11.	11. <u>Genus</u>	10
12.	12. <u>Species</u>	10
13.	13. <u>Common Name</u>	10
14.	14. <u>Family</u>	10
15.	15. <u>Genus</u>	10
16.	16. <u>Species</u>	10
17.	17. <u>Common Name</u>	10
18.	18. <u>Family</u>	10
19.	19. <u>Genus</u>	10
20.	20. <u>Species</u>	10
21.	21. <u>Common Name</u>	10
22.	22. <u>Family</u>	10
23.	23. <u>Genus</u>	10
24.	24. <u>Species</u>	10
25.	25. <u>Common Name</u>	10
26.	26. <u>Family</u>	10
27.	27. <u>Genus</u>	10
28.	28. <u>Species</u>	10
29.	29. <u>Common Name</u>	10
30.	30. <u>Family</u>	10
31.	31. <u>Genus</u>	10
32.	32. <u>Species</u>	10
33.	33. <u>Common Name</u>	10
34.	34. <u>Family</u>	10
35.	35. <u>Genus</u>	10
36.	36. <u>Species</u>	10
37.	37. <u>Common Name</u>	10
38.	38. <u>Family</u>	10
39.	39. <u>Genus</u>	10
40.	40. <u>Species</u>	10
41.	41. <u>Common Name</u>	10
42.	42. <u>Family</u>	10
43.	43. <u>Genus</u>	10
44.	44. <u>Species</u>	10
45.	45. <u>Common Name</u>	10
46.	46. <u>Family</u>	10
47.	47. <u>Genus</u>	10
48.	48. <u>Species</u>	10
49.	49. <u>Common Name</u>	10
50.	50. <u>Family</u>	10
51.	51. <u>Genus</u>	10
52.	52. <u>Species</u>	10
53.	53. <u>Common Name</u>	10
54.	54. <u>Family</u>	10
55.	55. <u>Genus</u>	10
56.	56. <u>Species</u>	10
57.	57. <u>Common Name</u>	10
58.	58. <u>Family</u>	10
59.	59. <u>Genus</u>	10
60.	60. <u>Species</u>	10
61.	61. <u>Common Name</u>	10
62.	62. <u>Family</u>	10
63.	63. <u>Genus</u>	10
64.	64. <u>Species</u>	10
65.	65. <u>Common Name</u>	10
66.	66. <u>Family</u>	10
67.	67. <u>Genus</u>	10
68.	68. <u>Species</u>	10
69.	69. <u>Common Name</u>	10
70.	70. <u>Family</u>	10
71.	71. <u>Genus</u>	10
72.	72. <u>Species</u>	10
73.	73. <u>Common Name</u>	10
74.	74. <u>Family</u>	10
75.	75. <u>Genus</u>	10
76.	76. <u>Species</u>	10
77.	77. <u>Common Name</u>	10
78.	78. <u>Family</u>	10
79.	79. <u>Genus</u>	10
80.	80. <u>Species</u>	10
81.	81. <u>Common Name</u>	10
82.	82. <u>Family</u>	10
83.	83. <u>Genus</u>	10
84.	84. <u>Species</u>	10
85.	85. <u>Common Name</u>	10
86.	86. <u>Family</u>	10
87.	87. <u>Genus</u>	10
88.	88. <u>Species</u>	10
89.	89. <u>Common Name</u>	10
90.	90. <u>Family</u>	10
91.	91. <u>Genus</u>	10
92.	92. <u>Species</u>	10
93.	93. <u>Common Name</u>	10
94.	94. <u>Family</u>	10
95.	95. <u>Genus</u>	10
96.	96. <u>Species</u>	10
97.	97. <u>Common Name</u>	10
98.	98. <u>Family</u>	10
99.	99. <u>Genus</u>	10
100.	100. <u>Species</u>	10

	<u>16A)</u>	
1) <u>Meer liebt Hörspiele</u>	10	22,5
2) <u>Das ist ein Witz</u>	10	22,5
3) <u>Was für eine Geschichte</u>	10	22,5
4) <u>Was kann ich machen?</u>	10	22,5
5) <u>Was ist das für ein Buch?</u>	10	22,5
6) <u>Was kann ich tun?</u>	10	22,5
7) <u>Was kann ich machen?</u>	10	22,5
8) <u>Was kann ich machen?</u>	10	22,5
9) <u>Was kann ich machen?</u>	10	22,5
10) <u>Was kann ich machen?</u>	10	22,5
11) <u>Was kann ich machen?</u>	10	22,5
12) <u>Was kann ich machen?</u>	10	22,5
13) <u>Was kann ich machen?</u>	10	22,5
14) <u>Was kann ich machen?</u>	10	22,5
15) <u>Was kann ich machen?</u>	10	22,5
16) <u>Was kann ich machen?</u>	10	22,5
17) <u>Was kann ich machen?</u>	10	22,5
18) <u>Was kann ich machen?</u>	10	22,5
19) <u>Was kann ich machen?</u>	10	22,5
20) <u>Was kann ich machen?</u>	10	22,5
21) <u>Was kann ich machen?</u>	10	22,5
22) <u>Was kann ich machen?</u>	10	22,5
23) <u>Was kann ich machen?</u>	10	22,5
24) <u>Was kann ich machen?</u>	10	22,5
25) <u>Was kann ich machen?</u>	10	22,5
26) <u>Was kann ich machen?</u>	10	22,5
27) <u>Was kann ich machen?</u>	10	22,5
28) <u>Was kann ich machen?</u>	10	22,5
29) <u>Was kann ich machen?</u>	10	22,5
30) <u>Was kann ich machen?</u>	10	22,5
31) <u>Was kann ich machen?</u>	10	22,5
32) <u>Was kann ich machen?</u>	10	22,5
33) <u>Was kann ich machen?</u>	10	22,5
34) <u>Was kann ich machen?</u>	10	22,5
35) <u>Was kann ich machen?</u>	10	22,5
36) <u>Was kann ich machen?</u>	10	22,5
37) <u>Was kann ich machen?</u>	10	22,5
38) <u>Was kann ich machen?</u>	10	22,5
39) <u>Was kann ich machen?</u>	10	22,5
40) <u>Was kann ich machen?</u>	10	22,5
41) <u>Was kann ich machen?</u>	10	22,5
42) <u>Was kann ich machen?</u>	10	22,5
43) <u>Was kann ich machen?</u>	10	22,5
44) <u>Was kann ich machen?</u>	10	22,5
45) <u>Was kann ich machen?</u>	10	22,5
46) <u>Was kann ich machen?</u>	10	22,5
47) <u>Was kann ich machen?</u>	10	22,5
48) <u>Was kann ich machen?</u>	10	22,5
49) <u>Was kann ich machen?</u>	10	22,5
50) <u>Was kann ich machen?</u>	10	22,5
51) <u>Was kann ich machen?</u>	10	22,5
52) <u>Was kann ich machen?</u>	10	22,5
53) <u>Was kann ich machen?</u>	10	22,5
54) <u>Was kann ich machen?</u>	10	22,5
55) <u>Was kann ich machen?</u>	10	22,5
56) <u>Was kann ich machen?</u>	10	22,5
57) <u>Was kann ich machen?</u>	10	22,5
58) <u>Was kann ich machen?</u>	10	22,5
59) <u>Was kann ich machen?</u>	10	22,5
60) <u>Was kann ich machen?</u>	10	22,5
61) <u>Was kann ich machen?</u>	10	22,5
62) <u>Was kann ich machen?</u>	10	22,5
63) <u>Was kann ich machen?</u>	10	22,5
64) <u>Was kann ich machen?</u>	10	22,5
65) <u>Was kann ich machen?</u>	10	22,5
66) <u>Was kann ich machen?</u>	10	22,5
67) <u>Was kann ich machen?</u>	10	22,5
68) <u>Was kann ich machen?</u>	10	22,5
69) <u>Was kann ich machen?</u>	10	22,5
70) <u>Was kann ich machen?</u>	10	22,5
71) <u>Was kann ich machen?</u>	10	22,5
72) <u>Was kann ich machen?</u>	10	22,5
73) <u>Was kann ich machen?</u>	10	22,5
74) <u>Was kann ich machen?</u>	10	22,5
75) <u>Was kann ich machen?</u>	10	22,5
76) <u>Was kann ich machen?</u>	10	22,5
77) <u>Was kann ich machen?</u>	10	22,5
78) <u>Was kann ich machen?</u>	10	22,5
79) <u>Was kann ich machen?</u>	10	22,5
80) <u>Was kann ich machen?</u>	10	22,5
81) <u>Was kann ich machen?</u>	10	22,5
82) <u>Was kann ich machen?</u>	10	22,5
83) <u>Was kann ich machen?</u>	10	22,5
84) <u>Was kann ich machen?</u>	10	22,5
85) <u>Was kann ich machen?</u>	10	22,5
86) <u>Was kann ich machen?</u>	10	22,5
87) <u>Was kann ich machen?</u>	10	22,5
88) <u>Was kann ich machen?</u>	10	22,5
89) <u>Was kann ich machen?</u>	10	22,5
90) <u>Was kann ich machen?</u>	10	22,5
91) <u>Was kann ich machen?</u>	10	22,5
92) <u>Was kann ich machen?</u>	10	22,5
93) <u>Was kann ich machen?</u>	10	22,5
94) <u>Was kann ich machen?</u>	10	22,5
95) <u>Was kann ich machen?</u>	10	22,5
96) <u>Was kann ich machen?</u>	10	22,5
97) <u>Was kann ich machen?</u>	10	22,5
98) <u>Was kann ich machen?</u>	10	22,5
99) <u>Was kann ich machen?</u>	10	22,5
100) <u>Was kann ich machen?</u>	10	22,5

```
In [6]: # Cell 2: Run the Loghi script  
%run xml_on_image_loghi.py
```



```
In [7]: 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 [8]: 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 [9]: xml_file_path = '/home/roderickmajoor/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 [10]: 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 [11]: 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 [12]: 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 [13]: 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 [14]: 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 [15]: 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 [16]: print("Jaccard Similarity:", jaccard_similarity(all_text_gt, filtered_list))
Jaccard Similarity: 0.4708029197080292
```

Now lets show the mean metrics for all images

```
In [17]: 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

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 [18]: 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 [19]: mean_wer, mean_cer, mean_matching_count, mean_jaccard_similarity = compute_m  
        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 do 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 value. This way, we can see the correct order. We will do this for the first image again.

```
In [20]: data_gt, data_loghi = main()
```

```
In [21]: 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 [22]: 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 [ ]:
```