

Redact Tool for ParanoiDF
Author: Patrick Wragg
25/07/2014
University of Kent
Supervisor: Julio Hernandez-
Castro



Purpose: To find word(s) behind a redaction box with high probability. Grammar parser included to narrow results to users preference.

Contents:

Page:

Introduction	3
Method One	4
Method Two	9
Grammar Parsing (Method Three)	14
Disclaimer	16

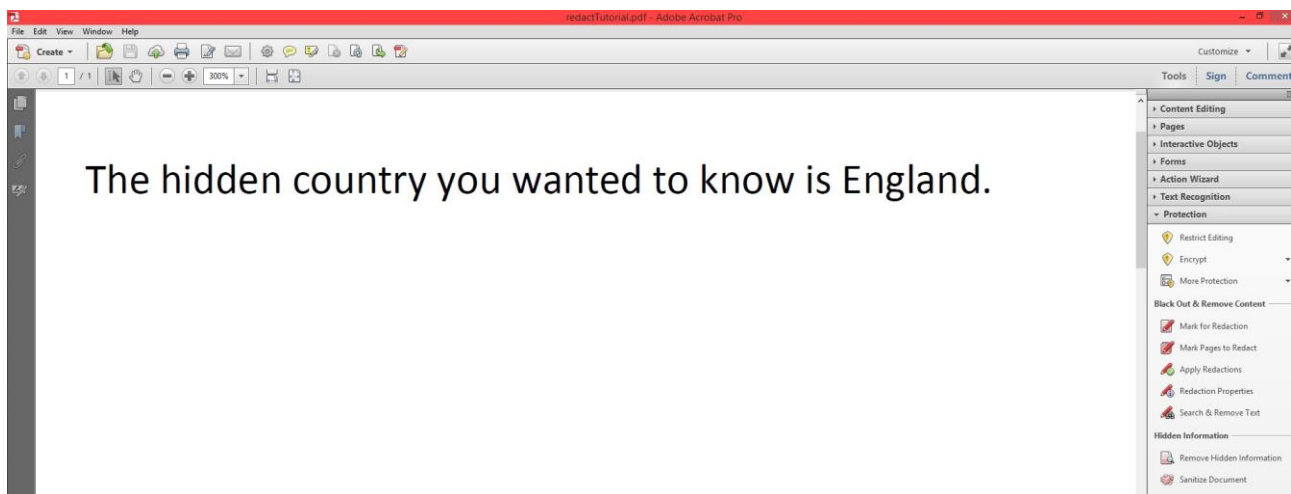
Introduction

Definition of Redaction: The censoring or obscuring of part of a text for legal or security purposes.

PLEASE READ THIS BEFORE CONTINUING:

- *There are several ways that PDFs structure text streams/redaction boxes, and it depends on the individual PDF (E.g. different versions).*
- *The fonts included with the tool are free ones obtained from www.fontpalace.com. Obviously I would break the law if I were to include those that are not free. Feel free to add any of your own fonts (free or paid for) to the “fonts” directory of the tool. They must be TrueType fonts (have the ext. “.ttf”)*
- *You have to do a small amount of word type deduction yourself (don’t be lazy) before running the command. **Reason:** There are about 150,000 different words in total, if you highly suspect the word is a country, why make it search through 150,000 when there are 196 countries?*
- *There are 3 pieces of information you need to collect before running the command “redact [option] [option]”: Font Size, Font, and Redaction box coordinates. The majority of this tutorial explains methods of how to find them. The actual command thereafter is fairly trivial.*
- *The more graphics/multimedia in a PDF, the longer it will take to identify the correct redact box object (“/BBox”). This is because the BBox (Bounding Box) object is used generically for most types of graphic/multimedia; it is not unique to the redaction box.*
- *Both the PDF’s for this tutorial were redacted securely using Adobe Acrobat Pro.*

Method One



The hidden country you wanted to know is XXXXXXXXXX.

After deducing that this word will most likely be a country, we can start:

Method 1:

1. Open the file using *"open input-file"*

```
ParanoIDF> open redactTutorial.pdf
File opened successfully!!

File: redactTutorial.pdf
MD5: 5e9eef3376745e5cd3e57c2b9c6dcccdf
SHA1: d16d2fd5748252dce3bcf3c69c93e5ba829d633b
Size: 24996 bytes
Version: 1.7
Binary: True
Linearized: True
Encrypted: False
Updates: 1
Objects: 26
Streams: 12
Comments: 0
Errors: 0

Version 0:
  Catalog: 10
  Info: No
  Objects (2): [9, 19]
  Streams (1): [19]
    Xref streams (1): [19]
    Encoded (1): [19]

Version 1:
  Catalog: 10
  Info: No
  Objects (24): [1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26]
  Compressed objects (11): [20, 21, 22, 23, 24, 25, 4, 5, 6, 7, 8]
  Streams (11): [26, 12, 13, 14, 15, 16, 17, 18, 1, 2, 3]
    Xref streams (1): [3]
    Object streams (3): [13, 1, 2]
    Encoded (11): [26, 12, 13, 14, 15, 16, 17, 18, 1, 2, 3]
  Suspicious elements:
    /AcroForm: [10]
    /OpenAction: [10]
```

2. The first thing I will do is search for /BBox (not case-sensitive): *"search /bbox"*. As you can see, two objects are returned, 17 and 18.

```
ParanoIDF> search /bbox

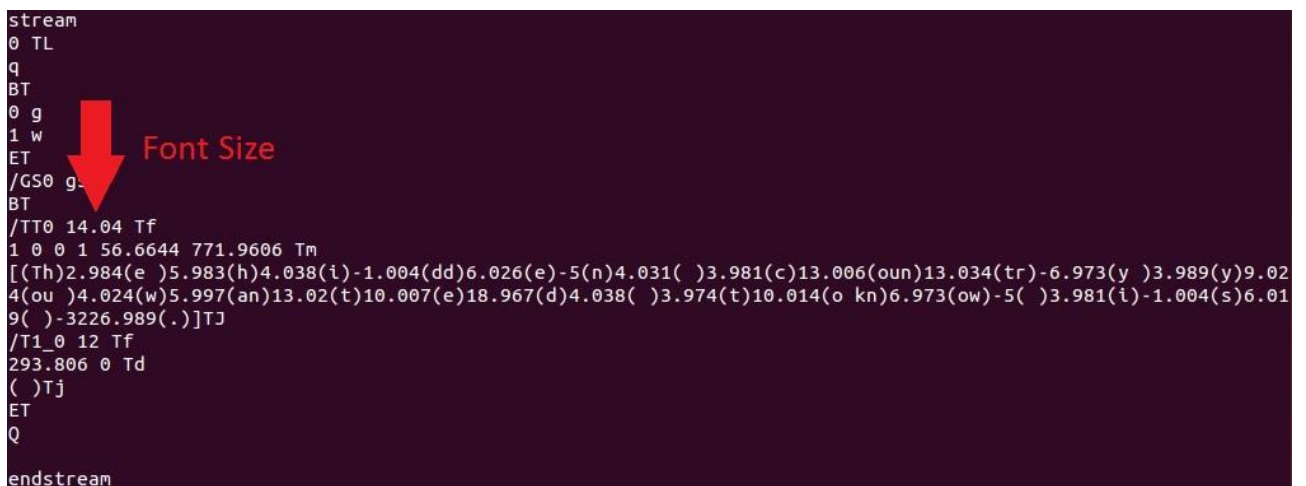
1: [17, 18]
```

3. I will now look at both objects. As you can see, object 17 is the text stream. This is handy, because the object containing the text stream should contain the font size. The font size is found just before **Tf** (Text Font Size) operator. In this case it is **14**. Note this down.

```
ParanoidPDF> object 17
<< /Filter /FlateDecode
/Resources << /ExtGState << /GS0 21 0 R >>
/Font << /TT0 25 0 R
/T1_0 23 0 R >>
/ProcSet [ /PDF /Text ] >>
/Length 246
/Group << /Type /Group
/S /Transparency >>
/Type /XObject
/FormType 1
/BBox [ 56.6644 769.462 353.471 783.961 ]
/Subtype /Form >>
stream
0 TL
q
BT
0 g
1 w
ET
/GS0 gs
BT
/TT0 14.04 Tf
1 0 0 1 56.6644 771.9606 Tm
[(Th)2.984(e )5.983(h)4.038(i)-1.004(dd)6.026(e)-5(n)4.031( )3.981(c)13.006(oun)13.034(tr)-6.973(y )3.989(y)9.02
4(ou )4.024(w)5.997(an)13.02(t)10.007(e)18.967(d)4.038( )3.974(t)10.014(o kn)6.973(ow)-5( )3.981(i)-1.004(s)6.01
9( )-3226.989(.)]Tj
/T1_0 12 Tf
293.806 0 Td
( )Tj
ET
Q
endstream
```

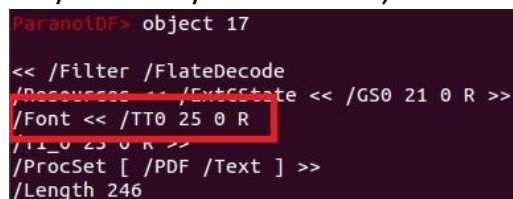


```
stream
0 TL
q
BT
0 g
1 w
ET
/GS0 gs
BT
/TT0 14.04 Tf
1 0 0 1 56.6644 771.9606 Tm
[(Th)2.984(e )5.983(h)4.038(i)-1.004(dd)6.026(e)-5(n)4.031( )3.981(c)13.006(oun)13.034(tr)-6.973(y )3.989(y)9.02
4(ou )4.024(w)5.997(an)13.02(t)10.007(e)18.967(d)4.038( )3.974(t)10.014(o kn)6.973(ow)-5( )3.981(i)-1.004(s)6.01
9( )-3226.989(.)]Tj
/T1_0 12 Tf
293.806 0 Td
( )Tj
ET
Q
endstream
```




4. Also in object 17 is the font object that is used for the text stream (this means it tells us the font used). It is located after the “/Font” operator. In this case it is “/Font << /TT0 25 0 R”; so the object is “25”.

```
ParanoidPDF> object 17
<< /Filter /FlateDecode
/Resources << /ExtGState << /GS0 21 0 R >>
/Font << /TT0 25 0 R
/T1_0 23 0 R >>
/ProcSet [ /PDF /Text ] >>
/Length 246
```



7. We should now go back to object 18 (found at step 2) now by doing “*object 18*”.

```
ParanoiDF> object 18
<< /Filter /FlateDecode
/BBBox [ 300.879 768.962 347.299 782.008 ]
/Resources << /ProcSet [ /PDF ] >>
/Length 72
/Type /XObject
/FormType 1
/Matrix [ 1.0 0.0 0.0 1.0 -300.879 -768.962 ]
/Subtype /Form >>
stream
0 g
1 0 0 1 0 0 cm
301.3793 769.4615 m
346.7986 769.4615 l
346.7986 781.5077 l
301.3793 781.5077 l
301.3793 769.4615 l
f
endstream
```

 **Redaction Box Coordinates**

8. In object 18 (above), we can deduct from comparing the two end coordinates of the text stream object 17, and this one, that this object represents the redact box. This is because one ends roughly near the other one. We will write these coordinates down: **300.879, 768.962, 347.299, 782.008**

9. Now we can finally do the command! “*redact c c*”. C and C are the two options I chose, because it is most likely a country, and is most likely a capitalised word.

Font Size = 14

Font = Calibri

Coordinates = 300.879, 768.962, 347.299, 782.008

```
ParanoiDF> redact c c
Enter 4 coordinates of BBox as shown in object: [n, n, n, n]
300.879
300.879, 768.962
300.879, 768.962, 347.299
300.879, 768.962, 347.299, 782.008
Coordinates: [300.879, 768.962, 347.299, 782.008]
X = 46 , Y = 13
Enter the font (Ext. not needed): Calibri
Enter the font size: 14
Parsing words now...
100%
Total matches: 24/197 words.
Results written to results.txt.
```


10. Done! After entering in the information and telling us that the redaction box size is 46x13 points, it outputted 24/197 successful words that fit the box. Below are the results, with the correct country "England" within the results.



Method Two:

Hello my name is ██████████

After deducing that this word will most likely be a name (Patrick), we can start:

1. Open the file.

```
ParanoiDF> open anotherRedactTutorial.pdf
File opened succesfully!!

File: anotherRedactTutorial.pdf
MD5: 028fcd1d25509d056896a3504becf8c0
SHA1: 22d5014146dd73259c39fe9ab54890f0a3332e39
Size: 88586 bytes
Version: 1.7
Binary: True
Linearized: True
Encrypted: False
Updates: 1
Objects: 32
Streams: 12
Comments: 0
Errors: 0

Version 0:
  Catalog: 20
  Info: 18
  Objects (2): [19, 26]
  Streams (1): [26]
    Xref streams (1): [26]
    Encoded (1): [26]

Version 1:
  Catalog: 20
  Info: 18
  Objects (30): [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 27, 28, 29, 30, 31, 32]
  Compressed objects (17): [27, 28, 29, 30, 31, 8, 9, 10, 7, 11, 12, 13, 14, 15, 16, 17, 18]
  Streams (11): [32, 22, 23, 24, 25, 1, 2, 3, 4, 5, 6]
    Xref streams (1): [6]
    Object streams (5): [23, 1, 2, 4, 5]
    Encoded (10): [32, 22, 23, 24, 25, 1, 2, 4, 5, 6]
  Suspicious elements:
    /AcroForm: [20]
```

2. The first thing I did here was search for /BBox by doing “search /bbox”. As you can see, object 25 is the result.

```
ParanoiDF> search /bbox
1: [25]
```

3. Using the command “*object 25*”, I will look inside the object.

```
ParanoiDF> object 25

<< /Filter /FlateDecode
/BBBox [ 150.57 756.22 182.25 768.26 ]
/Resources << /ProcSet [ /PDF ] >>
/Length 68
/Type /XObject
/FormType 1
/Matrix [ 1.0 0.0 0.0 1.0 -150.57 -756.22 ]
/Subtype /Form >>
stream
0 g
1 0 0 1 0 0 cm
151.0701 756.72 m
181.7504 756.72 l
181.7504 767.76 l
151.0701 767.76 l
151.0701 756.72 l
f
endstream
```

4. Seeing as it seems like it doesn't have any useful information, I will search for whether it has any references. I will do the command “*references to 25*” to see if there are any objects that are connected to this object. This object references 21.

```
ParanoiDF> references to 25

[21]
```

5. I am now looking inside object 21 “*object 21*”. This object catches my eye as it has a “/Font” operator which points to an object.

```
ParanoiDF> object 21

<< /Contents 22 0 R
/Parent 17 0 R
/Tabs /S
/Resources << /XObject << /Fm0 25 0 R >>
/Font << /TT0 31 0 R >>
/ProcSet [ /PDF /Text ] >>
/Rotate 0
/CropBox [ 0 0 595.32 841.92 ]
/Group << /CS /DeviceRGB
/S /Transparency
/Type /Group >>
/MediaBox [ 0 0 595.32 841.92 ]
/Type /Page
/StructParents 0 >>
```

6. I will now check the other objects that are in object 21. At the top there is “/Contents 22 0 R”. I will check this one. Result! (Below) This object (22) holds the text stream (which means the font size too). I write down the font size number, which is the digit before “Tf”, in this case, it is **11**. It is a good idea to take note of the coordinate number it has given below so we can use it to help identify which object is the redaction box.

```
ParanoIDF> object 22
<< /Length 179
/Filter /FlateDecode >>
stream
BT
/P.../Lang (en-GB)/MCTD...>>BDC
/TT0 11.04 Tf
72.021 753.18 Td
[(H)2.998(ello)-2.998( )9.004(m)-4.004(y)7.002( nam)7.002(e )-3.007(is )-2767.139( )]TJ
EMC
ET
q
1 0 0 1 149.5701141 755.2199707
/Fm0 Do
Q
endstream
```

Font size

Take note

7. Going back to object 21, I look inside the object that is next to “/Font << /TT0 31 0 R”, so object 31. Object 31 has the font in it. In this case it is **Calibri**

```
ParanoIDF> object 31
<< /FirstChar 32
/Widths 30 0 R
/Encoding /WinAnsiEncoding
/Type /Font
/BaseFont /ABCDEE+Calibri
/LastChar 121
/FontDescriptor 29 0 R
/Subtype /TrueType >>
```

8. After going back to object 25 (step 2), I deduce that this must be the redaction box because its coordinates are similar to the text stream one I took note of earlier. The redaction box coordinates are: **150.57, 756.22, 182.25, 768.26**

```
ParanoidF> object 25
Coordinates
<< /Filter /FlateDecode
/BBox [ 150.57 756.22 182.25 768.26 ]
/Resources << /ProcSet [ /PDF ] >>
/Length 68
/Type /XObject
/FormType 1
/Matrix [ 1.0 0.0 0.0 1.0 -150.57 -756.22 ]
/Subtype /Form >>
stream
0 g
1 0 0 1 0 0 cm
151.0701 756.72 m
181.7504 756.72 l
181.7504 767.76 l
151.0701 767.76 l
151.0701 756.72 l
f
endstream
```

9. Now we can finally do the command! “*redact f c*”. I used F and C because it was most likely to be a first name, and capitalised.

Font Size = 11

Font = Calibri

Coordinates = 150.57 756.22 182.25 768.26

```
ParanoidF> redact f c
Enter 4 coordinates of BBox as shown in object: [n, n, n, n]
150.57
150.57, 756.22
150.57, 756.22, 182.25
150.57, 756.22, 182.25, 768.26
Coordinates: [150.57, 756.22, 182.25, 768.26]
X = 31 , Y = 12
Enter the font (Ext. not needed): Calibri
Enter the font size: 11
Parsing words now...
100%
Total matches: 1061/5494 words.
Results written to /home/test/Desktop/ParanoidF/results.txt.
```

10. Done! After entering in the information and telling us that the redaction box size is 31x12 points, it outputted 1,061/5,494 successful words that fit the box. Below are the results, with the correct country "England" within the results.



Grammar Parsing (Method Three)

After checking a word, you may want to do a parsing of the grammar of the sentence that the word is in to refine the results based on grammatical parse-ability score.

1. After retrieving the successful words, it will ask you if you want to do a grammar check. Type in “y” to proceed.

```
student@csvm2C30:~/Desktop$ python ParanoiDF/paranoiDF.py -i
ParanoiDF> redact w u
Enter 4 coordinates of BBox as shown in object: [n, n, n, n]
150.57
150.57, 756.22
150.57, 756.22, 182.25
150.57, 756.22, 182.25, 768.26
Coordinates: [150.57, 756.22, 182.25, 768.26]
X = 31 , Y = 12
Enter the font (Ext. not needed): Calibri
Enter the font size: 11
Parsing words now...
100%
Total matches: 5899/58110 words.

Would you like to do a grammar check? y/n:
```

```
Would you like to do a grammar check? y/n: ☐
```

2. You now have to add the sentence that the word was in, where “\$word” is the wildcard for where the word is situated in the sentence. It can be anywhere in the sentence. The example I have chosen is “*The \$word crossed the road.*”, so if the word is “people”, the sentence it parses is: “*The people crossed the road.*”.

```
Enter sentence where $word is word: The $word crossed the road.
```


3. The parser will now load all of the sentences from a temporary file.

```
Loading parser from serialized file edu/stanford/nlp/models/lexparser/englishPCFG.ser.gz ... done [3.0 sec].
Parsing file: /home/student/Desktop/ParanoiDF/tempSentence.txt
```

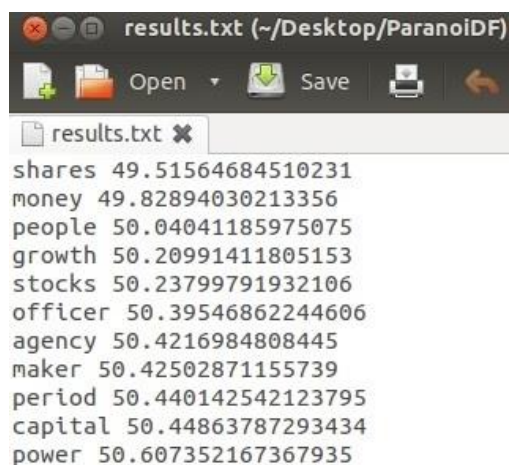
```
Parsing [sent. 1179 len. 7]: The critical crossed the road . .
Parsing [sent. 1180 len. 7]: The critter crossed the road . .
Parsing [sent. 1181 len. 7]: The croaks crossed the road . .
Parsing [sent. 1182 len. 7]: The croatia crossed the road . .
Parsing [sent. 1183 len. 7]: The crocus crossed the road . .
Parsing [sent. 1184 len. 7]: The crofter crossed the road . .
Parsing [sent. 1185 len. 7]: The crones crossed the road . .
Parsing [sent. 1186 len. 7]: The crooks crossed the road . .
Parsing [sent. 1187 len. 7]: The croons crossed the road . .
Parsing [sent. 1188 len. 7]: The crosier crossed the road . .
Parsing [sent. 1189 len. 7]: The crossly crossed the road . .
Parsing [sent. 1190 len. 7]: The crouch crossed the road . .
Parsing [sent. 1191 len. 7]: The crowds crossed the road . .
Parsing [sent. 1192 len. 7]: The crowns crossed the road . .
Parsing [sent. 1193 len. 7]: The crozier crossed the road . .
Parsing [sent. 1194 len. 7]: The crucial crossed the road . .
Parsing [sent. 1195 len. 7]: The crucifix crossed the road . .
```

4. Once it is done, it will display this message.

```
Parsed file: /home/student/Desktop/ParanoiDF/tempSentence.txt [5899 sentences].
Parsed 41294 words in 5899 sentences (239.11 wds/sec; 34.16 sents/sec).
```

5. Now all you have to do is specify how many results you want it to output to "results.txt". It will sort the results so that the best parsed sentences will be at the top, and the worst will be at the bottom (lower the score the better).

```
Number of results? 100
Results written to /home/student/Desktop/ParanoiDF/results.txt.
```



```
shares 49.51564684510231
money 49.82894030213356
people 50.04041185975075
growth 50.20991411805153
stocks 50.23799791932106
officer 50.39546862244606
agency 50.4216984808445
maker 50.42502871155739
period 50.440142542123795
capital 50.44863787293434
power 50.607352167367935
```


Disclaimer:

- *The results are not perfect. It is designed to find the word behind the redaction box based on high probability.*
- *The grammar parsing is not an indication of whether the sentence makes “human grammatical sense”. It is an indication of how well it parses based on a number of factors. The lower the score, the better parse-ability score it receives (see below).*

From question 17 of <http://www-nlp.stanford.edu/software/parser-faq.shtml>:

“Why does the parser accept incorrect/ungrammatical sentences?”

This parser is in the space of modern statistical parsers whose goal is to give the most likely sentence analysis to a list of words. It does not attempt to determine grammaticality, though it will normally prefer a "grammatical" parse for a sentence if one exists. This is appropriate in many circumstances, such as when wanting to interpret user input, or dealing with conversational speech, web pages, non-native speakers, etc.

For other applications, such as grammar checking, this is less appropriate. One could attempt to assess grammaticality by looking at the probabilities that the parser returns for sentences, but it is difficult to normalize this number to give a useful "grammaticality" score, since the probability strongly depends on other factors like the length of the sentence, the rarity of the words in the sentence, and whether word dependencies in the sentence being tested were seen in the training data or not.“