Methods for digital forensics

Gavin Sonne

Computer Science, Hartnell College, Salinas, CA 93901

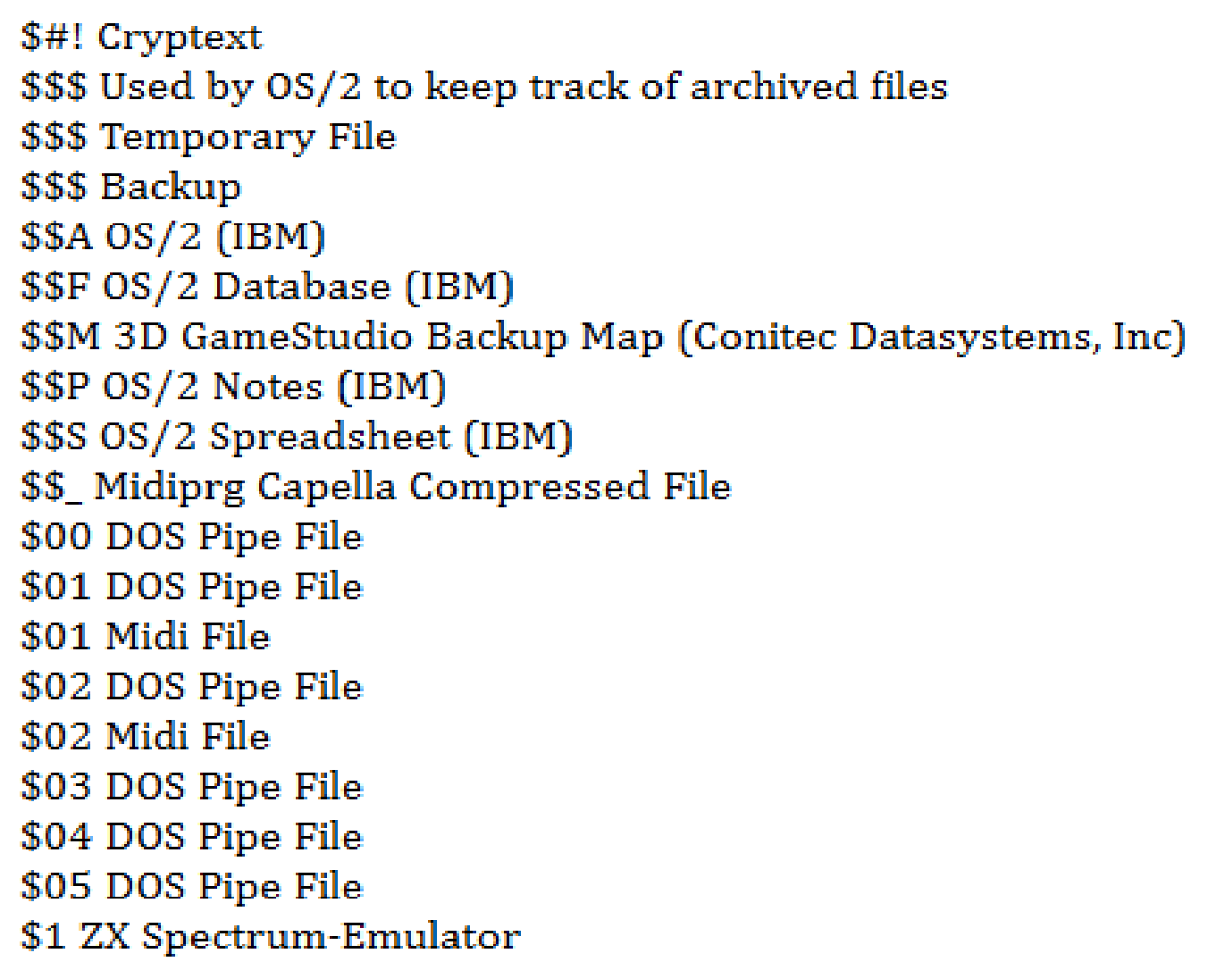
Neil Rowe, Professor, DEEP Lab

Abstract

The purpose of this project was to improve the existing methods used by Neil Rowe and the DEEP (Digital Exploration and Exploitaition) Group for classifying file extensions, top level directories, and bottom level directories. For classifying file extensions and directories, A numbering scheme is used to classify different types. To classify them, a combination of python programs and personal evaluation and user input was used. Once the file extension decoder python script was run on the list of file extensions and the classification key, the file extensions were classified individually, line by line, into the numbering scheme. The same method was used to classify top level and bottom level directories, though there was no classification key for these. Without a classification key, these had to be classified either by common sense, or by using Google. To conclude this project, Neil Rowe verified that all the classifications were correct, and added this new classified data to the existing scheme to classify file types and directories.

Materials and methods

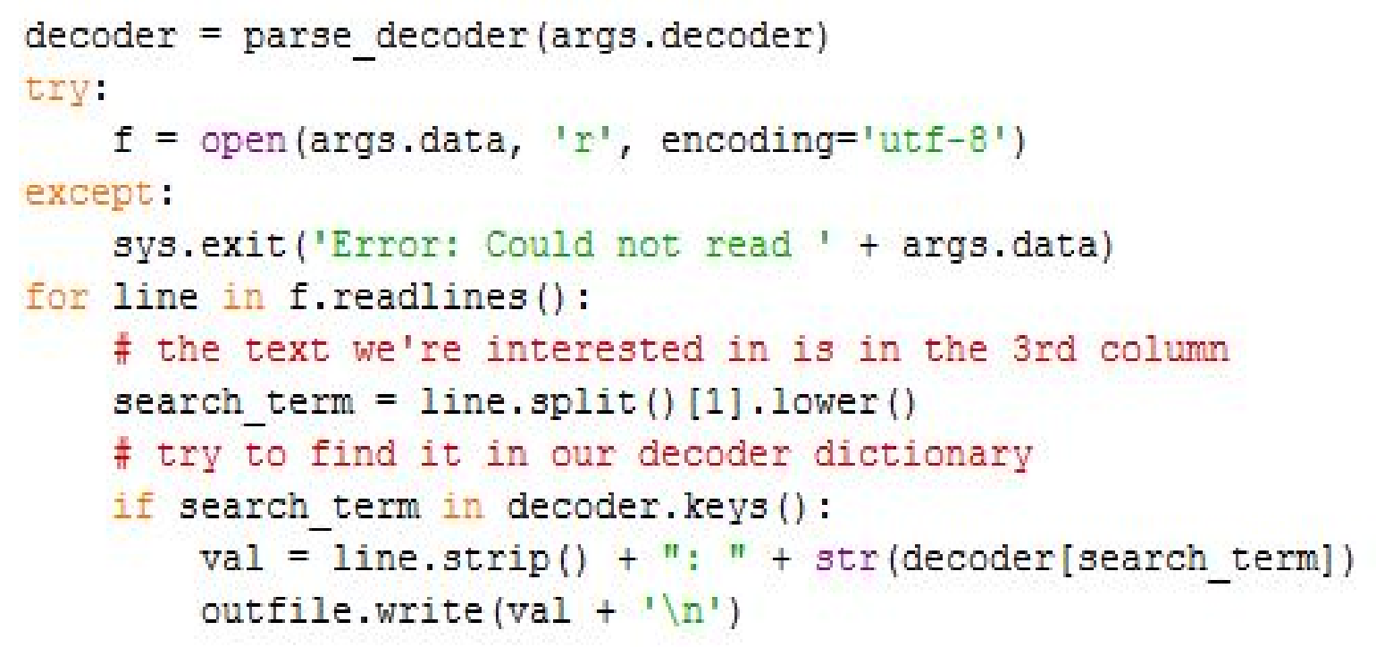
First, the file extension descriptions were cut and pasted from the websites filext.com and file-extensions.org into a text file.



```
$#! Cryptext
$$$ Used by OS/2 to keep track of archived files
$$$ Temporary File
$$$ Backup
$$A OS/2 (IBM)
$$F OS/2 Database (IBM)
$$M 3D GameStudio Backup Map (Conitec Datasystems, Inc)
$$P OS/2 Notes (IBM)
$$S OS/2 Spreadsheet (IBM)
$$_ Midiprg Capella Compressed File
$00 DOS Pipe File
$01 DOS Pipe File
$01 Midi File
$02 DOS Pipe File
$02 Midi File
$03 DOS Pipe File
$04 DOS Pipe File
$05 DOS Pipe File
$1 ZX Spectrum-Emulator
```

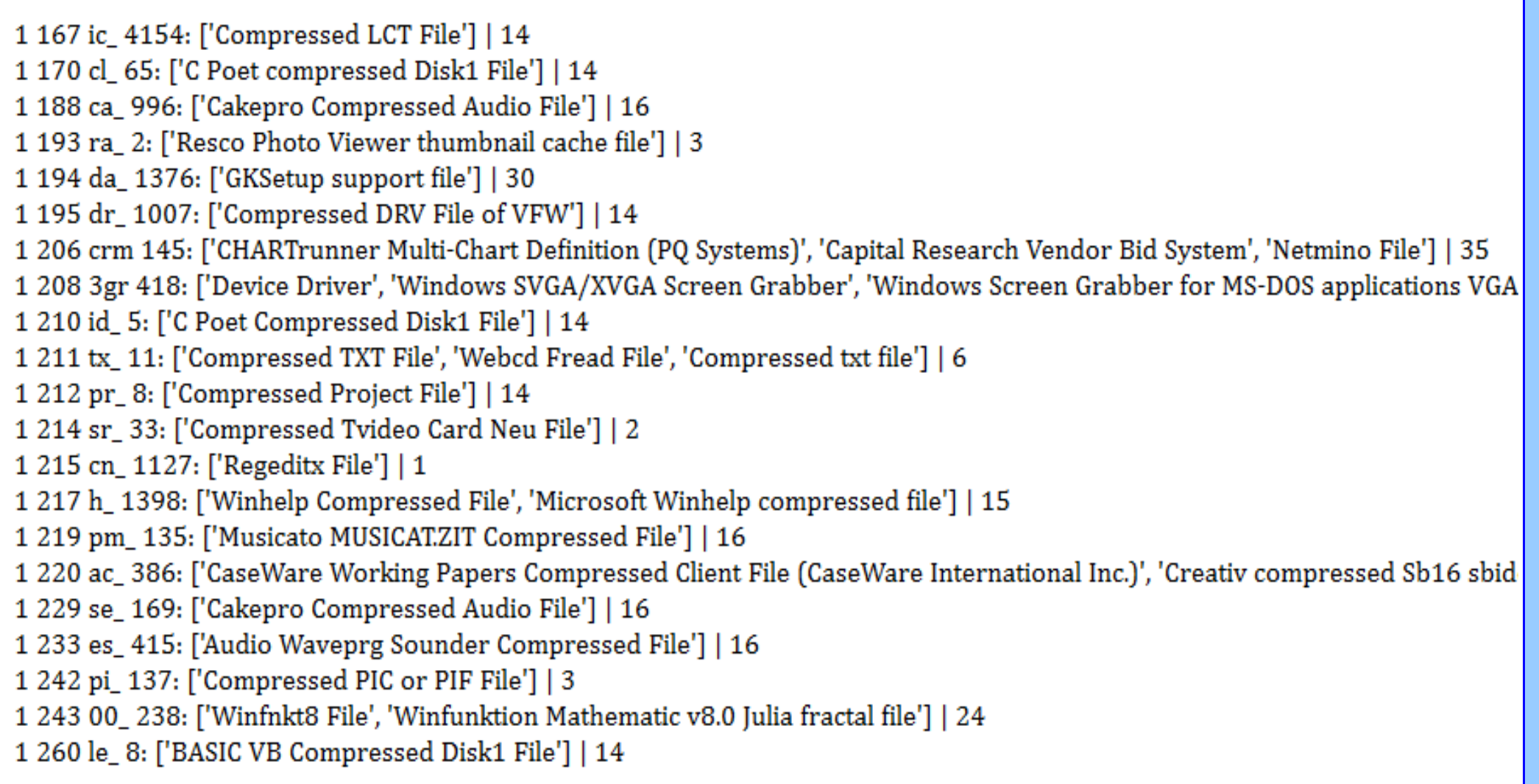
Fig 1. Sample of the descriptions text file.

Next, a python script was used to join these file extension descriptions with the file extension list from our corpus.



```python
decoder = parse_decoder(args.decoder)
try:
    f = open(args.data, 'r', encoding='utf-8')
except:
    sys.exit('Error: Could not read ' + args.data)
for line in f.readlines():
    # the text we're interested in is in the 3rd column
    search_term = line.split()[1].lower()
    # try to find it in our decoder dictionary
    if search_term in decoder.keys():
        val = line.strip() + ": " + str(decoder[search_term])
        outfile.write(val + '\n')
```

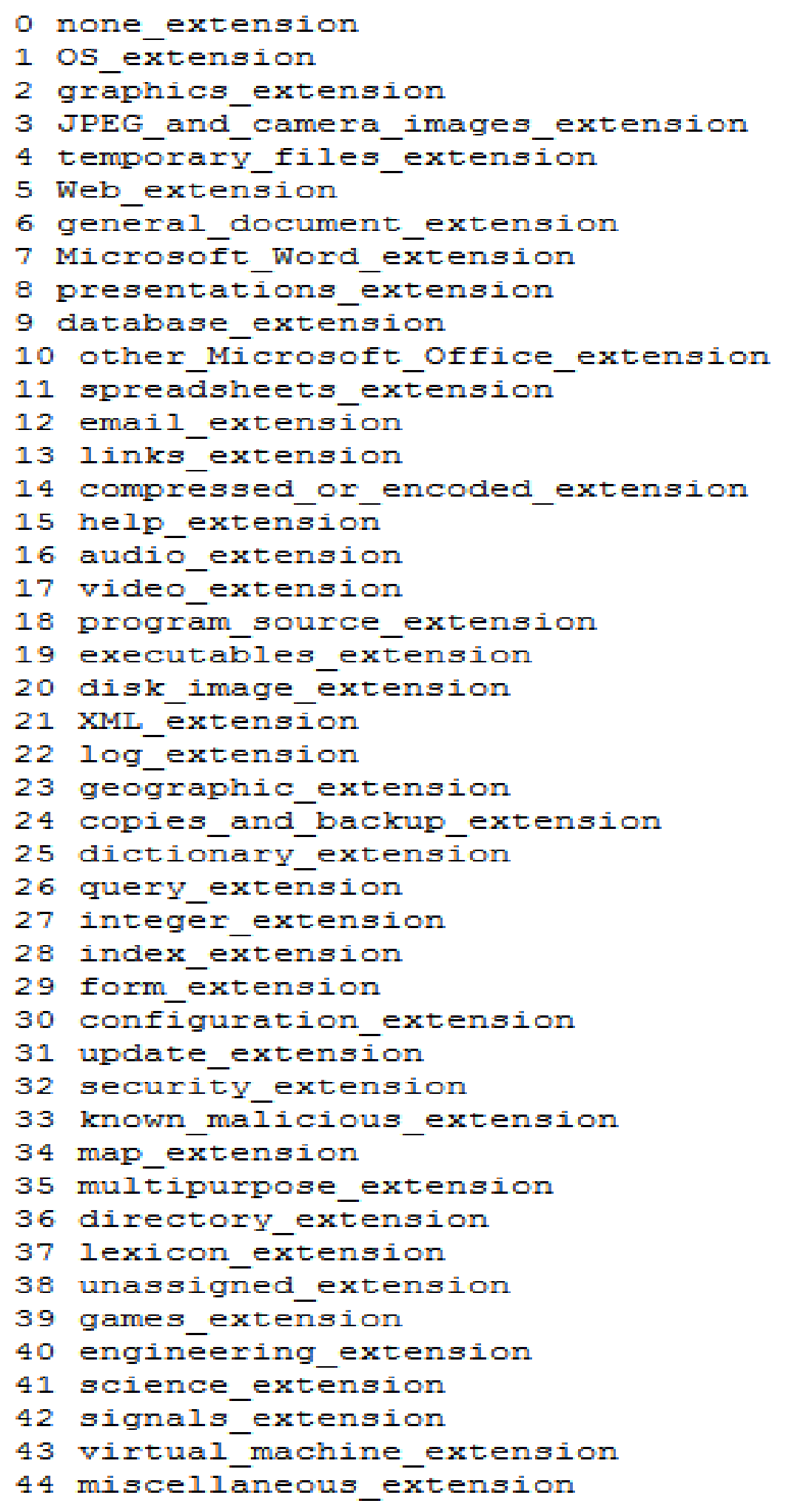
Fig 2. Sample of the decoder Python script



```
1 167 ic_ 4154: ['Compressed LCT File'] | 14
1 170 cl_ 65: ['C Poet compressed Disk1 File'] | 14
1 188 ca_ 996: ['Cakepro Compressed Audio File'] | 16
1 193 ra_ 2: ['Resco Photo Viewer thumbnail cache file'] | 3
1 194 da_ 1376: ['GKSetup support file'] | 30
1 195 dr_ 1007: ['Compressed DRV File of VFW'] | 14
1 206 crm 145: ['CHARTrunner Multi-Chart Definition (PQ Systems)', 'Capital Research Vendor Bid System', 'Netmino File'] | 35
1 208 3gr 418: ['Device Driver', 'Windows SVGA/XVGA Screen Grabber', 'Windows Screen Grabber for MS-DOS applications VGA
1 210 id_ 5: ['C Poet Compressed Disk1 File'] | 14
1 211 tx_ 11: ['Compressed TXT File', 'Webcd Fread File', 'Compressed txt file'] | 6
1 212 pr_ 8: ['Compressed Project File'] | 14
1 214 sr_ 33: ['Compressed Tvideo Card Neu File'] | 2
1 215 cn_ 1127: ['Regeditx File'] | 1
1 217 h_ 1398: ['Winhelp Compressed File', 'Microsoft Winhelp compressed file'] | 15
1 219 pm_ 135: ['Musicato MUSICAT.ZIT Compressed File'] | 16
1 220 ac_ 386: ['CaseWare Working Papers Compressed Client File (CaseWare International Inc.)', 'Creativ compressed Sb16 sbid
1 229 se_ 169: ['Cakepro Compressed Audio File'] | 16
1 233 es_ 415: ['Audio Waveprg Sounder Compressed File'] | 16
1 242 pi_ 137: ['Compressed PIC or PIF File'] | 3
1 243 00_ 238: ['Winfnkt8 File', 'Winfunktion Mathematic v8.0 Julia fractal file'] | 24
1 260 le_ 8: ['BASIC VB Compressed Disk1 File'] | 14
```

Fig 3. Sample of the extension and descriptions joined and classified

Finally, the extensions are classified with Neil Rowe's numbering scheme, one by one.



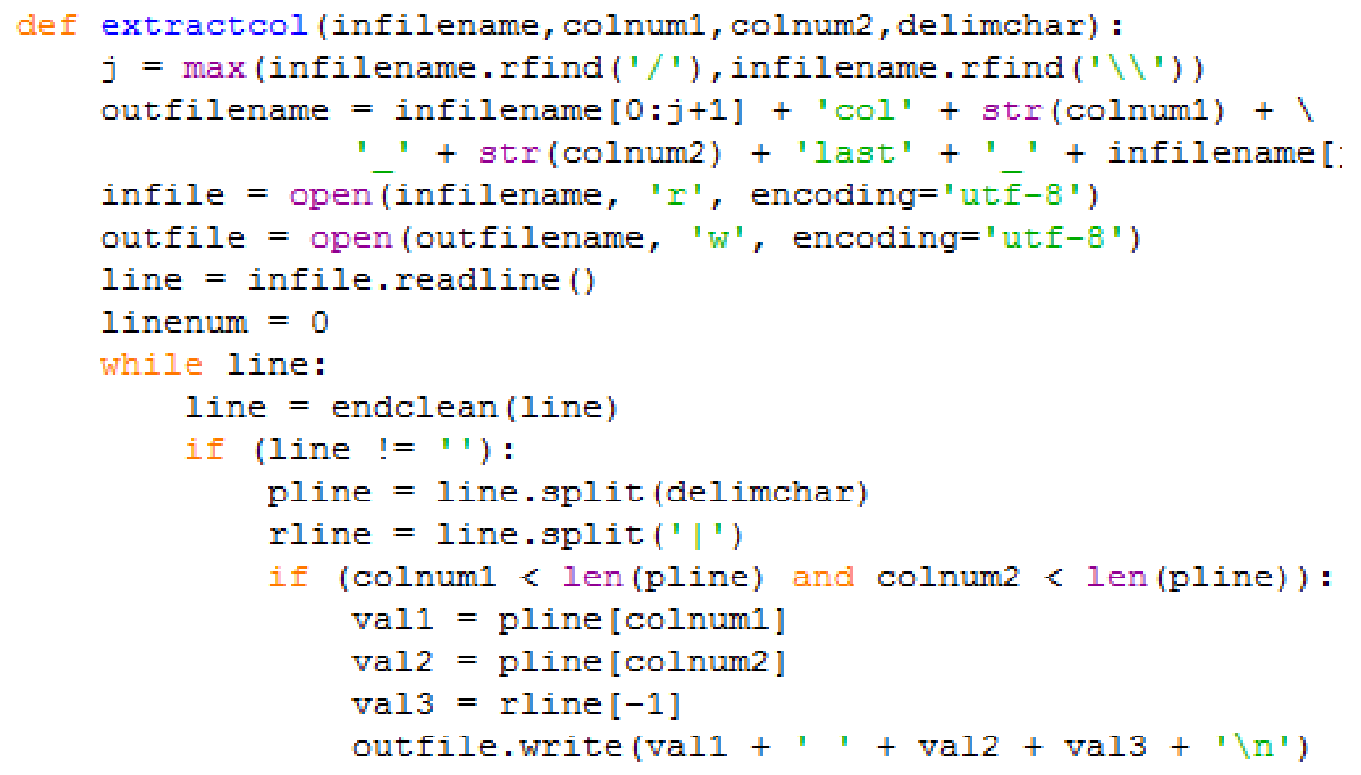
```
0 none_extension
1 OS_extension
2 graphics_extension
3 JPEG_and_camera_images_extension
4 temporary_files_extension
5 Web_extension
6 general_document_extension
7 Microsoft_Word_extension
8 presentations_extension
9 database_extension
10 other_Microsoft_Office_extension
11 spreadsheets_extension
12 email_extension
13 links_extension
14 compressed_or_encoded_extension
15 help_extension
16 audio_extension
17 video_extension
18 program_source_extension
19 executables_extension
20 disk_image_extension
21 XML_extension
22 log_extension
23 geographic_extension
24 copies_and_backup_extension
25 dictionary_extension
26 query_extension
27 integer_extension
28 index_extension
29 form_extension
30 configuration_extension
31 update_extension
32 security_extension
33 known_malicious_extension
34 map_extension
35 multipurpose_extension
36 directory_extension
37 lexicon_extension
38 unassigned_extension
39 games_extension
40 engineering_extension
41 science_extension
42 signals_extension
43 virtual_machine_extension
44 miscellaneous_extension
```

Fig 4. Big Subset Mappings text file, used to classify the extensions.

For top level and bottom level directories, the same methods were used, except there was no good online list of these, so they were all classified manually, using either Google or the context of the directory names.

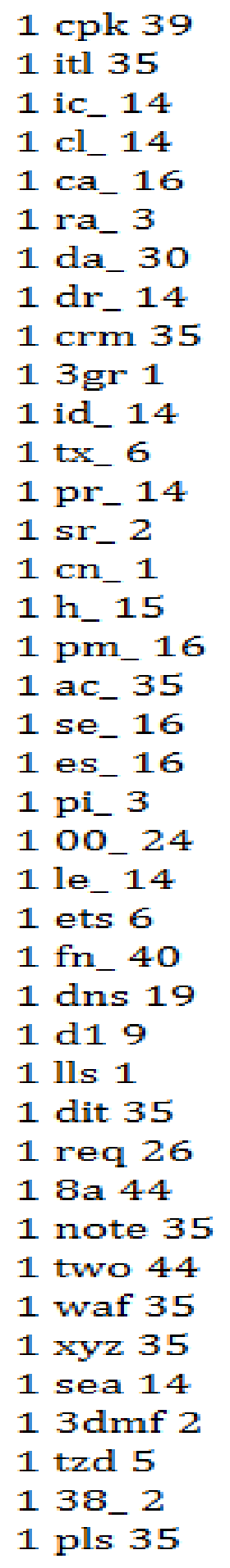
Results

Once this section had been completed, which involved classifying the file extension and directories line by line (for file extensions this was 5077 lines, and took a week of work), the text file containing the classified file extensions was stripped of the descriptions, as they are not needed for the classification scheme.



```python
def extractcol(infilename,colnum1,colnum2,delimchar):
    j = max(infilename.rfind('/'),infilename.rfind('\\'))
    outfilename = infilename[0:j+1] + 'col' + str(colnum1) + \
                  '_' + str(colnum2) + 'last' + '_' + infilename[
    infile = open(infilename, 'r', encoding='utf-8')
    outfile = open(outfilename, 'w', encoding='utf-8')
    line = infile.readline()
    linenum = 0
    while line:
        line = endclean(line)
        if (line != ''):
            pline = line.split(delimchar)
            rline = line.split('|')
            if (colnum1 < len(pline) and colnum2 < len(pline)):
                val1 = pline[colnum1]
                val2 = pline[colnum2]
                val3 = rline[-1]
                outfile.write(val1 + ' ' + val2 + val3 + '\n')
```

Fig 5. Sample of the Python script to extract columns



```
1 cpk 39
1 itl 35
1 ic_ 14
1 cl_ 14
1 ca_ 16
1 ra_ 3
1 da_ 30
1 dr_ 14
1 crm 35
1 3gr 1
1 id_ 14
1 tx_ 6
1 pr_ 14
1 sr_ 2
1 cn_ 1
1 h_ 15
1 pm_ 16
1 ac_ 35
1 se_ 16
1 es_ 16
1 pi_ 3
1 00_ 24
1 le_ 14
1 ets 6
1 fn_ 40
1 dns 19
1 d1 9
1 lls 1
1 dit 35
1 req 26
1 8a 44
1 note 35
1 two 44
1 waf 35
1 xyz 35
1 sea 14
1 3dmf 2
1 tzd 5
1 38_ 2
1 pls 35
```

Fig 6. Sample of the final, classified file.

Conclusions

With the project completed, Neil Rowe went over the classifications to make sure they were all correct. With that done, they were added to the Big Subset Mappings text file, to aid in better classification of file types.

This work will improve the algorithms being used to classify file types, top level directories, and bottom level directories. When the classification script is run, the results will be much more accurate.

Acknowledgments

Naval Postgraduate School:

Gavin Sonne, Intern

Neil Rowe, Professor

Joel Young, Associate Professor

Riqui Schwamm, Masters Student

This internship was funded by Hartnell College.

Fig 7. Gavin Sonne (Intern) and Neil Rowe (Professor)

Gavin Sonne plans on attending Hartnell College for one more year, before transferring to a CSU or UC to complete a Bachelors of Science in Computer Science.

For further information

Please contact ncrowe@nps.edu for further information, or visit his website at http://faculty.nps.edu/ncrowe/