# Automatic analysis of text files

Lior Shamir

Lawrence technological University

## Contents

## Synopsis

The purpose of this document is to show how to perform automatic classification and analysis of text files. Automatic classification of text files is done by computers "reading" the files automatically. In its basic form, classification of text files using machine learning is performed by first converting each text files to a set of numerical values that describes it. Then the computer program identifies repetitive patterns in these numbers and uses these patterns to automatically classify or annotate these text files.

In our experiment we will use text data taken from social media such as Facebook or Twitter, and use them to find links between the way people use social media and other indicators such as demographic characteristics.

Automatic text analysis is an increasingly growing field of research, with immediate application to marketing, sales, branding, and more, driven by the ability of sophisticated algorithms to analyze large databases of posts produced by the public and communicated through social media. It can also be used for basic research for understanding how and why social media is used.

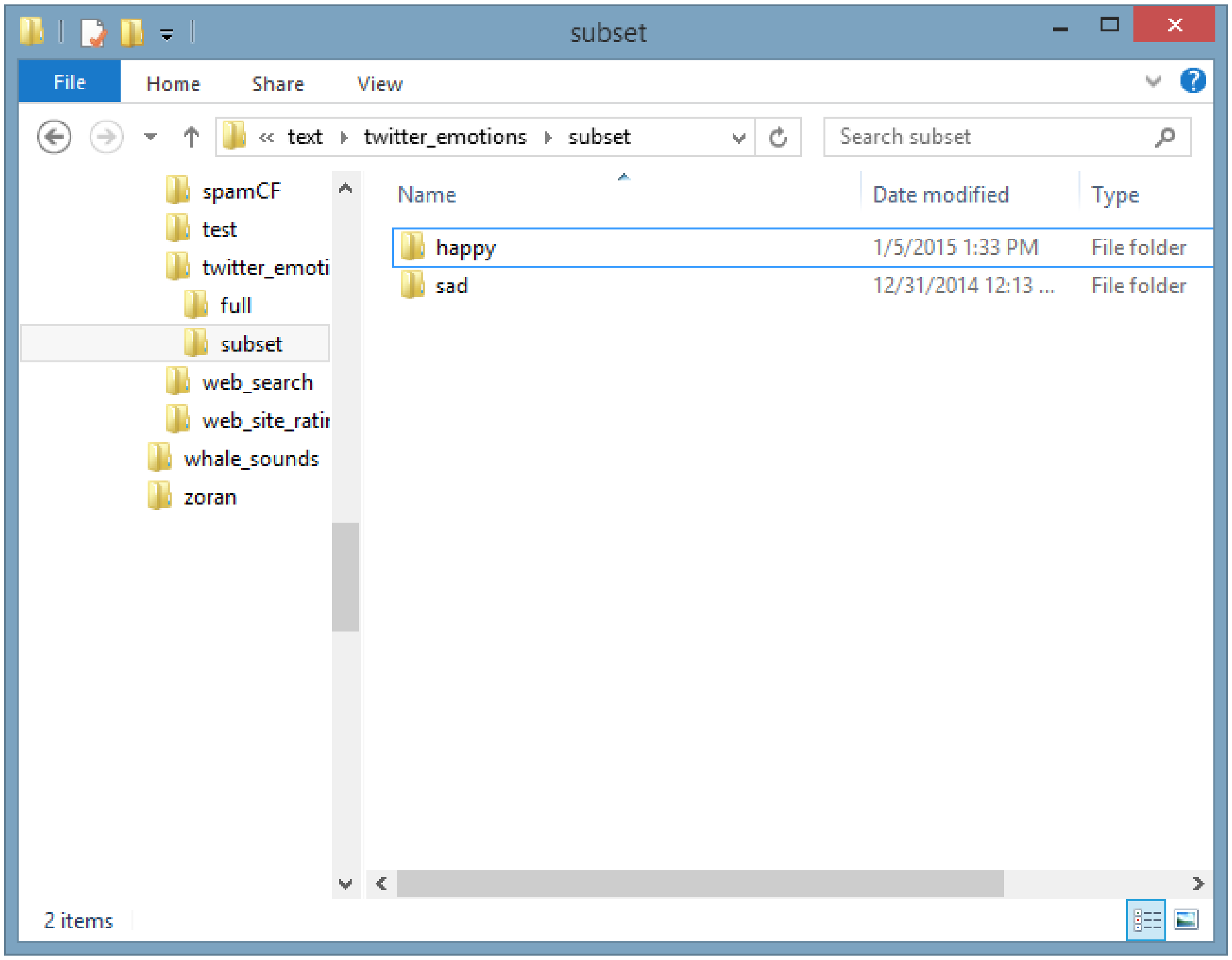
## 1. Organizing the text files

There are two options to organize the text files: By folders or by an association file. Before starting create a folder for the experiment in your hard drive. Such folder will contain all the elements necessary for the text analysis.

### 1.1. Organize text files by folders

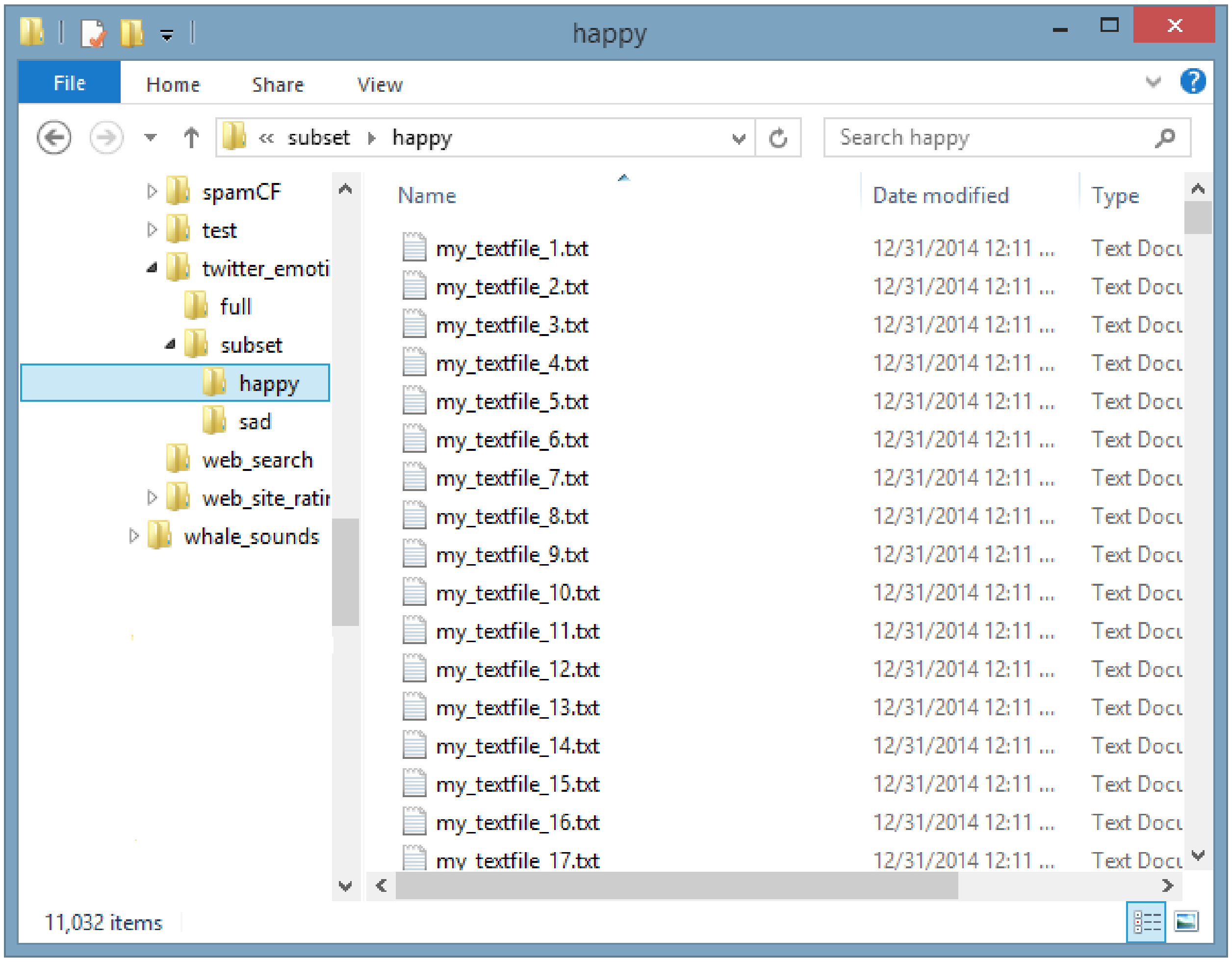
The organization of the text files in folders is very important for the experiment, and the processing of the files depends on the correct organization of the files in folders. The number of folders and the type of folder can vary and depends on the goal of the experiment. For example, if we want to investigate gender differences in Facebook, we could create two folders, one for male posts and one for female posts in which we will sort text files (tweets) according to the gender of the twitter user. The text files should be saved in the hard drive such that each group of files is saved in a different folder. Saving the files in a fashion that does not follow the correct folder structure might require substantial efforts in re-organizing the files after they are downloaded and stored.

As a concrete example, let’s consider the following experiment: Suppose that we want to test whether emotions tweets (happy or sad) are associated with the length of the words used in the tweet. In this case we would need to create two categories – happy and sad, create a folder for each of the categories, and copy text files of happy and sad twitter posts into these folders. The folder structure would be the following:



| Name | Date modified | Type |
|---|---|---|
| happy | 1/5/2015 1:33 PM | File folder |
| sad | 12/31/2014 12:13 ... | File folder |

The folder "happy" should contain all text files of happy twitter posts, and the folder "sad" should contain all text files of sad twitter posts. For instance, the folder happy would look like the following:



| Name | Date modified | Type |
|---|---|---|
| my_textfile_1.txt | 12/31/2014 12:11 ... | Text Docu |
| my_textfile_2.txt | 12/31/2014 12:11 ... | Text Docu |
| my_textfile_3.txt | 12/31/2014 12:11 ... | Text Docu |
| my_textfile_4.txt | 12/31/2014 12:11 ... | Text Docu |
| my_textfile_5.txt | 12/31/2014 12:11 ... | Text Docu |
| my_textfile_6.txt | 12/31/2014 12:11 ... | Text Docu |
| my_textfile_7.txt | 12/31/2014 12:11 ... | Text Docu |
| my_textfile_8.txt | 12/31/2014 12:11 ... | Text Docu |
| my_textfile_9.txt | 12/31/2014 12:11 ... | Text Docu |
| my_textfile_10.txt | 12/31/2014 12:11 ... | Text Docu |
| my_textfile_11.txt | 12/31/2014 12:11 ... | Text Docu |
| my_textfile_12.txt | 12/31/2014 12:11 ... | Text Docu |
| my_textfile_13.txt | 12/31/2014 12:11 ... | Text Docu |
| my_textfile_14.txt | 12/31/2014 12:11 ... | Text Docu |
| my_textfile_15.txt | 12/31/2014 12:11 ... | Text Docu |
| my_textfile_16.txt | 12/31/2014 12:11 ... | Text Docu |
| my_textfile_17.txt | 12/31/2014 12:11 ... | Text Docu |

## 1.2. Organize the text files using an association file

Another option is to create, using notepad or another text editor, a text file that associates each twitter post text file to its specific class or value (in the example case, happy or sad). In such reference file, each line contains a full path to the text file, a space (or tab) and the value. In our example there are just two possible values: 1 represent a happy post and 2 represent a sad twitter post.

For instance, the file can be the following:

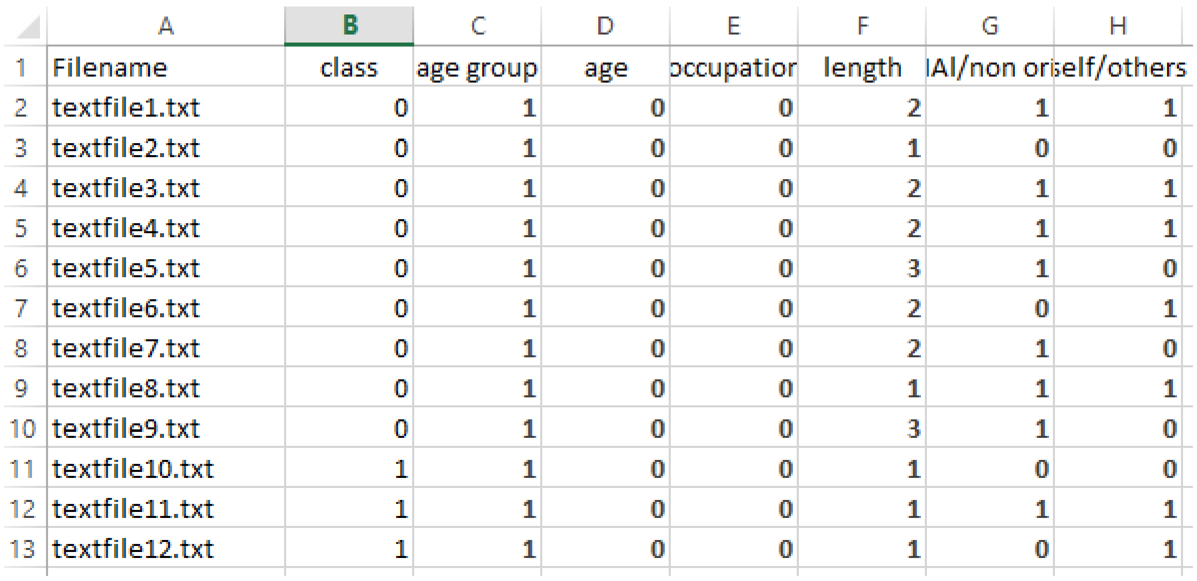
```
C:\data\twitter_emotions\subset\happy\my_textfile_11.txt	1
C:\data\twitter_emotions\subset\happy\my_textfile_12.txt	1
C:\data\twitter_emotions\subset\happy\my_textfile_13.txt	1
C:\data\twitter_emotions\subset\happy\my_textfile_14.txt	1
C:\data\twitter_emotions\subset\happy\my_textfile_15.txt	1
C:\data\twitter_emotions\subset\sad\my_textfile_26.txt	2
C:\data\twitter_emotions\subset\sad\my_textfile_27.txt	2
C:\data\twitter_emotions\subset\sad\my_textfile_28.txt	2
C:\data\twitter_emotions\subset\sad\my_textfile_29.txt	2
```

For instance, the text file “my_textfile_11.txt” contains the text of a happy twitter post, and therefore labeled “1”. The file “my_textfile_26.txt” contains a sad tweet, and is therefore labelled as “2”. The order of the files is not important.

### 1.2.2. Creating the text file from data in Excel spreadsheet

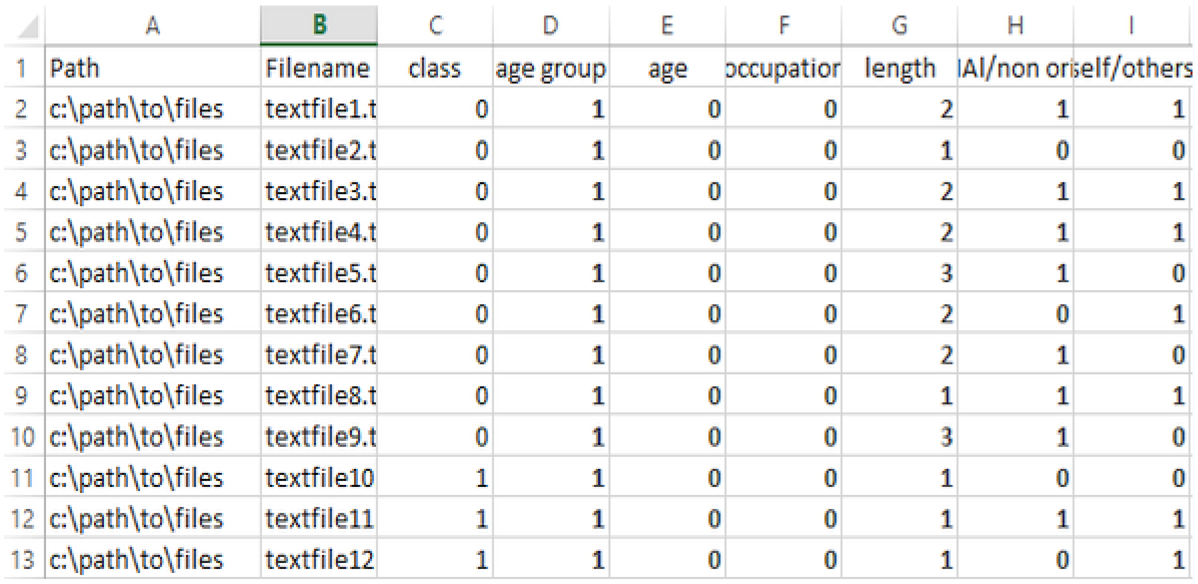
The text file can also be created from an Excel spreadsheet:

Suppose the spreadsheet is arranged such that one column has the filename of the text file, and the other columns are values related to the file as the following example:



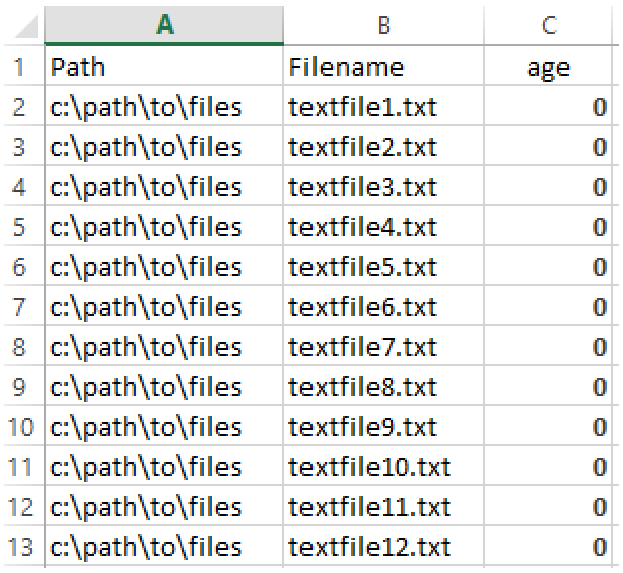
|  | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | Filename | class | age group | age | occupatior | length | IAl/non or | self/others |
| 2 | textfile1.txt | 0 | 1 | 0 | 0 | 2 | 1 | 1 |
| 3 | textfile2.txt | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 4 | textfile3.txt | 0 | 1 | 0 | 0 | 2 | 1 | 1 |
| 5 | textfile4.txt | 0 | 1 | 0 | 0 | 2 | 1 | 1 |
| 6 | textfile5.txt | 0 | 1 | 0 | 0 | 3 | 1 | 0 |
| 7 | textfile6.txt | 0 | 1 | 0 | 0 | 2 | 0 | 1 |
| 8 | textfile7.txt | 0 | 1 | 0 | 0 | 2 | 1 | 0 |
| 9 | textfile8.txt | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 10 | textfile9.txt | 0 | 1 | 0 | 0 | 3 | 1 | 0 |
| 11 | textfile10.txt | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 12 | textfile11.txt | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 13 | textfile12.txt | 1 | 1 | 0 | 0 | 1 | 0 | 1 |

Suppose we want to analyze how the age of the poster correlates with the content of a Facebook or twitter post. First, we need to align all cells to the right and add a column with the path to the folders where the text files are located. It is important that the path does not have spaces.



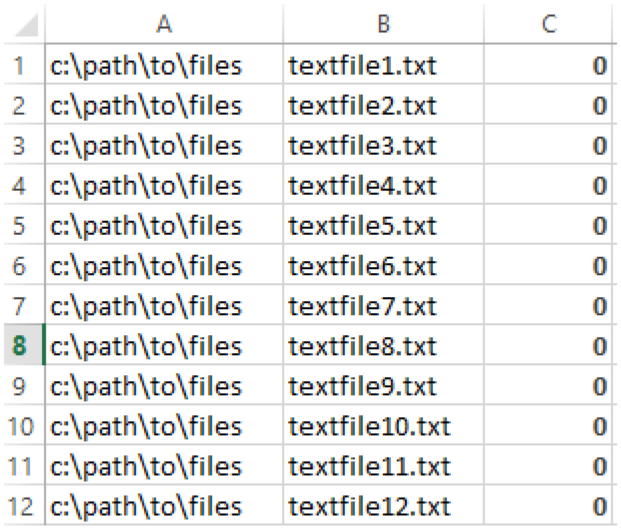
| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Path | Filename | class | age group | age | occupatior | length | IAl/non or | self/others |
| 2 | c:\path\to\files | textfile1.t | 0 | 1 | 0 | 0 | 2 | 1 | 1 |
| 3 | c:\path\to\files | textfile2.t | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 4 | c:\path\to\files | textfile3.t | 0 | 1 | 0 | 0 | 2 | 1 | 1 |
| 5 | c:\path\to\files | textfile4.t | 0 | 1 | 0 | 0 | 2 | 1 | 1 |
| 6 | c:\path\to\files | textfile5.t | 0 | 1 | 0 | 0 | 3 | 1 | 0 |
| 7 | c:\path\to\files | textfile6.t | 0 | 1 | 0 | 0 | 2 | 0 | 1 |
| 8 | c:\path\to\files | textfile7.t | 0 | 1 | 0 | 0 | 2 | 1 | 0 |
| 9 | c:\path\to\files | textfile8.t | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 10 | c:\path\to\files | textfile9.t | 0 | 1 | 0 | 0 | 3 | 1 | 0 |
| 11 | c:\path\to\files | textfile10 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 12 | c:\path\to\files | textfile11 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 13 | c:\path\to\files | textfile12 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |

Now, the column of interest should be moved to the right of the column of the file name, and the other columns should be removed.



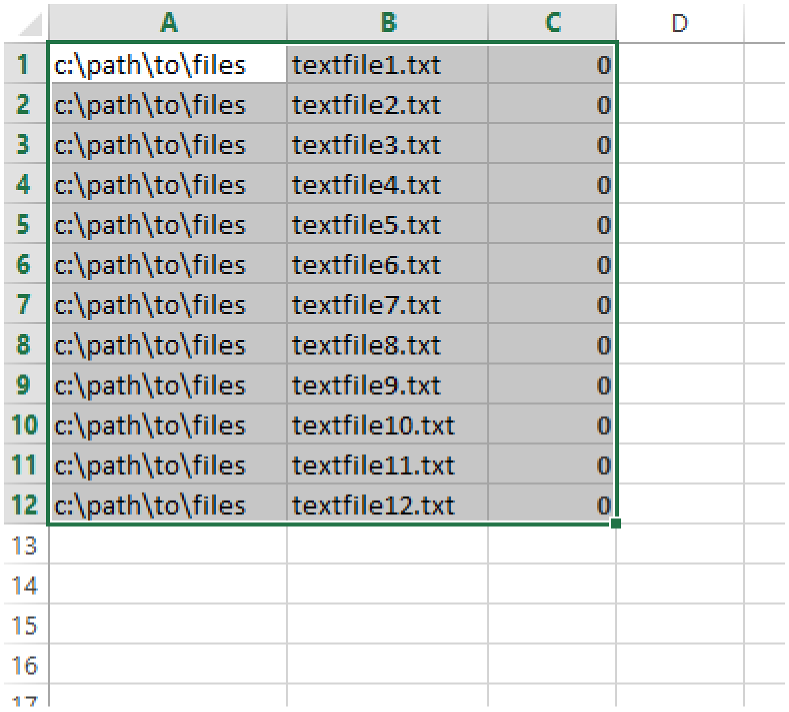
| | A | B | C |
|---|---|---|---|
| 1 | Path | Filename | age |
| 2 | c:\path\to\files | textfile1.txt | 0 |
| 3 | c:\path\to\files | textfile2.txt | 0 |
| 4 | c:\path\to\files | textfile3.txt | 0 |
| 5 | c:\path\to\files | textfile4.txt | 0 |
| 6 | c:\path\to\files | textfile5.txt | 0 |
| 7 | c:\path\to\files | textfile6.txt | 0 |
| 8 | c:\path\to\files | textfile7.txt | 0 |
| 9 | c:\path\to\files | textfile8.txt | 0 |
| 10 | c:\path\to\files | textfile9.txt | 0 |
| 11 | c:\path\to\files | textfile10.txt | 0 |
| 12 | c:\path\to\files | textfile11.txt | 0 |
| 13 | c:\path\to\files | textfile12.txt | 0 |

The column titles should also be removed.



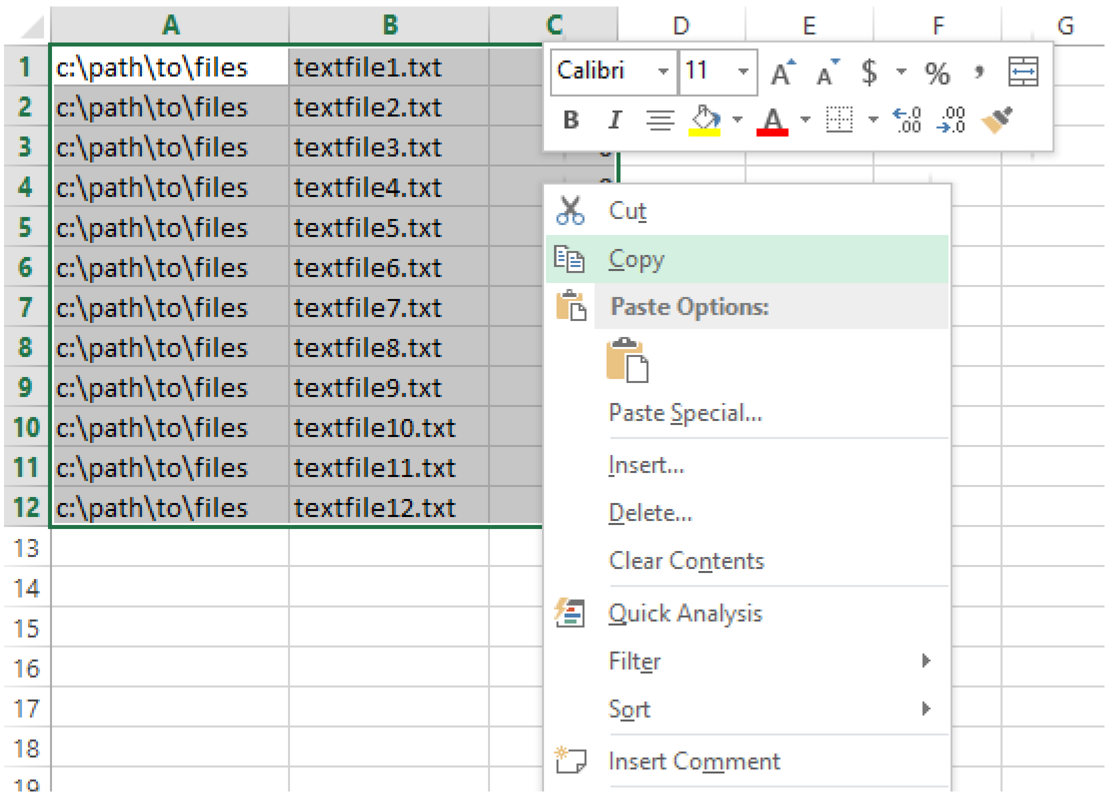
| | A | B | C |
|---|---|---|---|
| 1 | c:\path\to\files | textfile1.txt | 0 |
| 2 | c:\path\to\files | textfile2.txt | 0 |
| 3 | c:\path\to\files | textfile3.txt | 0 |
| 4 | c:\path\to\files | textfile4.txt | 0 |
| 5 | c:\path\to\files | textfile5.txt | 0 |
| 6 | c:\path\to\files | textfile6.txt | 0 |
| 7 | c:\path\to\files | textfile7.txt | 0 |
| 8 | c:\path\to\files | textfile8.txt | 0 |
| 9 | c:\path\to\files | textfile9.txt | 0 |
| 10 | c:\path\to\files | textfile10.txt | 0 |
| 11 | c:\path\to\files | textfile11.txt | 0 |
| 12 | c:\path\to\files | textfile12.txt | 0 |

Highlight the entire data with the mouse.



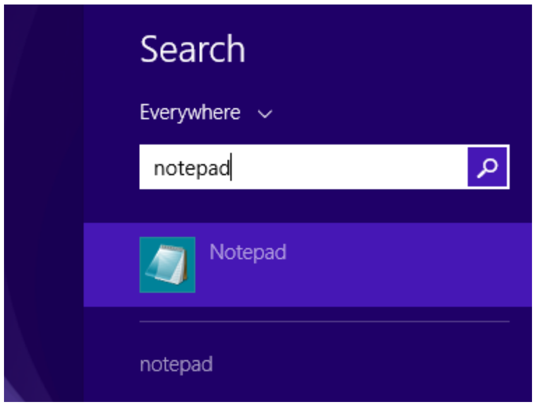
| | A | B | C | D |
|---|---|---|---|---|
| 1 | c:\path\to\files | textfile1.txt | 0 | |
| 2 | c:\path\to\files | textfile2.txt | 0 | |
| 3 | c:\path\to\files | textfile3.txt | 0 | |
| 4 | c:\path\to\files | textfile4.txt | 0 | |
| 5 | c:\path\to\files | textfile5.txt | 0 | |
| 6 | c:\path\to\files | textfile6.txt | 0 | |
| 7 | c:\path\to\files | textfile7.txt | 0 | |
| 8 | c:\path\to\files | textfile8.txt | 0 | |
| 9 | c:\path\to\files | textfile9.txt | 0 | |
| 10 | c:\path\to\files | textfile10.txt | 0 | |
| 11 | c:\path\to\files | textfile11.txt | 0 | |
| 12 | c:\path\to\files | textfile12.txt | 0 | |
| 13 | | | | |
| 14 | | | | |
| 15 | | | | |
| 16 | | | | |

And then right-click the mouse and select “Copy”

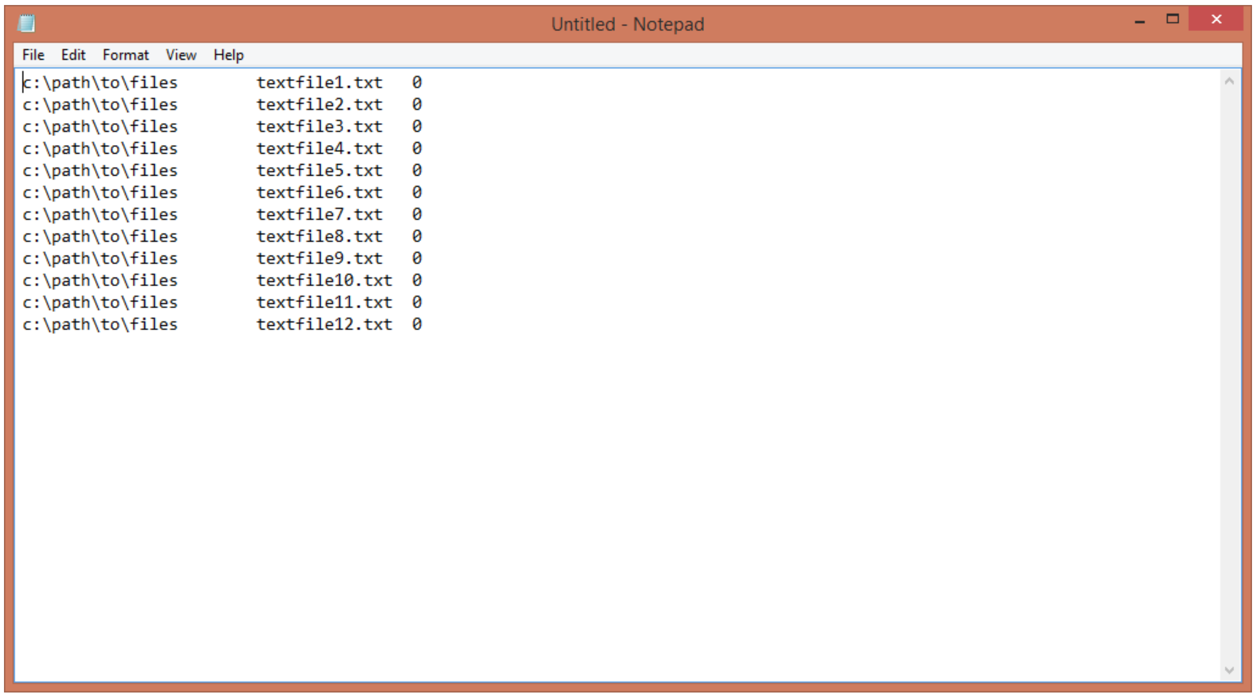


| | A | B |
|---|---|---|
| 1 | c:\path\to\files | textfile1.txt |
| 2 | c:\path\to\files | textfile2.txt |
| 3 | c:\path\to\files | textfile3.txt |
| 4 | c:\path\to\files | textfile4.txt |
| 5 | c:\path\to\files | textfile5.txt |
| 6 | c:\path\to\files | textfile6.txt |
| 7 | c:\path\to\files | textfile7.txt |
| 8 | c:\path\to\files | textfile8.txt |
| 9 | c:\path\to\files | textfile9.txt |
| 10 | c:\path\to\files | textfile10.txt |
| 11 | c:\path\to\files | textfile11.txt |
| 12 | c:\path\to\files | textfile12.txt |

Now click the Windows button at the bottom left corner and then type "notepad", and then press <enter> to launch Notepad.

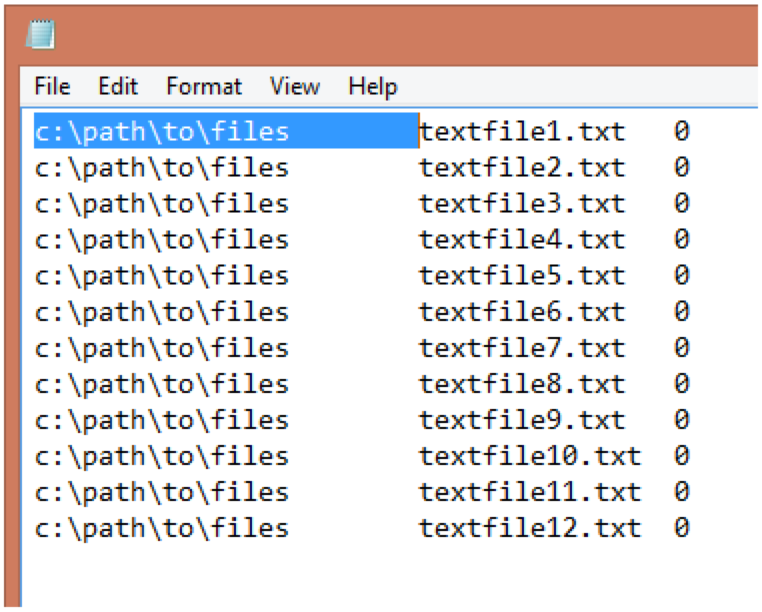


In Notepad, click the "Edit" menu and then click "Paste", or just press the ctrl-V to paste the data from Excel to notepad. Notepad should show the following file:



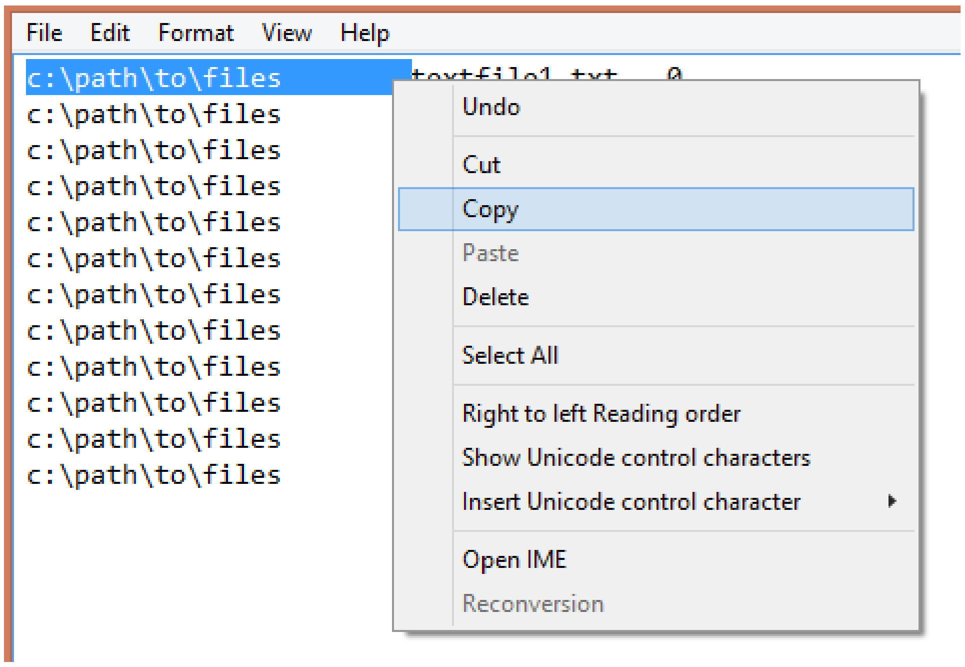
```
c:\path\to\files        textfile1.txt   0
c:\path\to\files        textfile2.txt   0
c:\path\to\files        textfile3.txt   0
c:\path\to\files        textfile4.txt   0
c:\path\to\files        textfile5.txt   0
c:\path\to\files        textfile6.txt   0
c:\path\to\files        textfile7.txt   0
c:\path\to\files        textfile8.txt   0
c:\path\to\files        textfile9.txt   0
c:\path\to\files        textfile10.txt  0
c:\path\to\files        textfile11.txt  0
c:\path\to\files        textfile12.txt  0
```

To replace the spaces between the path and the file name, you first need to highlight the path including the spaces until the name of the file.



```
c:\path\to\files        textfile1.txt   0
c:\path\to\files        textfile2.txt   0
c:\path\to\files        textfile3.txt   0
c:\path\to\files        textfile4.txt   0
c:\path\to\files        textfile5.txt   0
c:\path\to\files        textfile6.txt   0
c:\path\to\files        textfile7.txt   0
c:\path\to\files        textfile8.txt   0
c:\path\to\files        textfile9.txt   0
c:\path\to\files        textfile10.txt  0
c:\path\to\files        textfile11.txt  0
c:\path\to\files        textfile12.txt  0
```

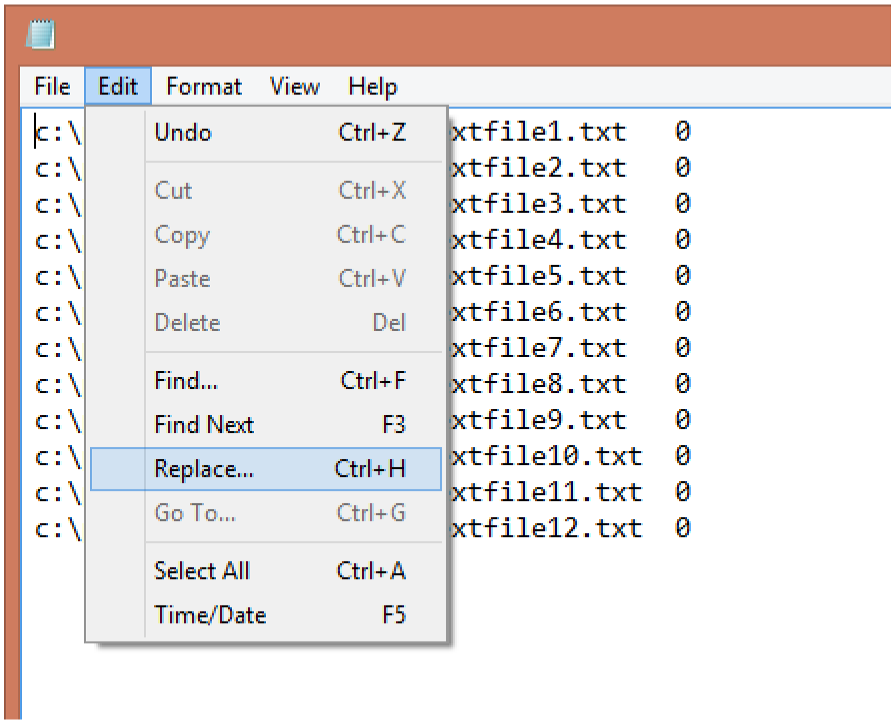
Then right-click and select “Copy”.



```
c:\path\to\files        textfile1.txt    0
c:\path\to\files
c:\path\to\files
c:\path\to\files
c:\path\to\files
c:\path\to\files
c:\path\to\files
c:\path\to\files
c:\path\to\files
c:\path\to\files
c:\path\to\files
c:\path\to\files
```

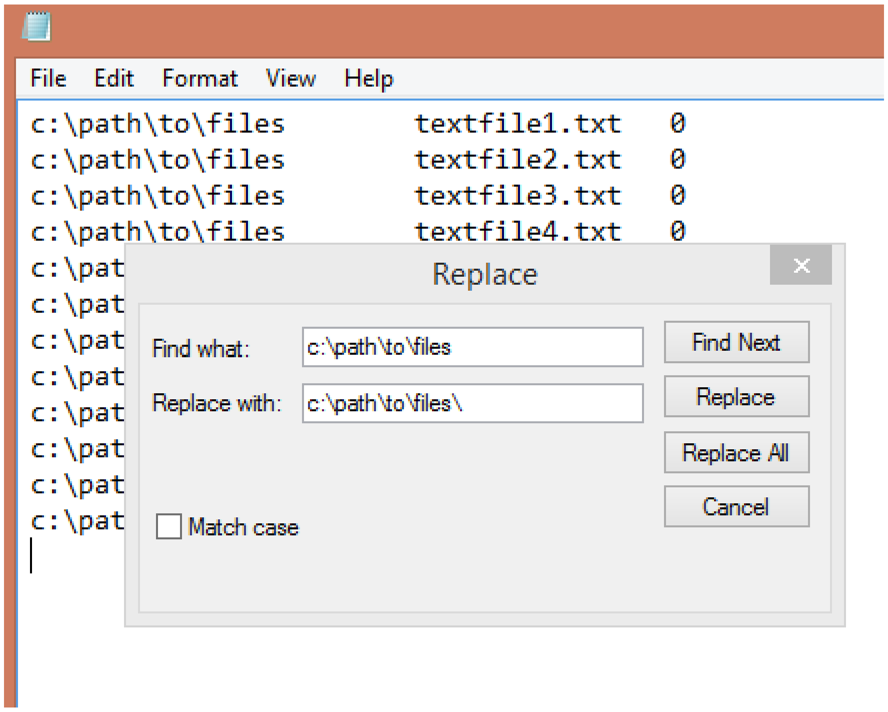
File
Edit
Format
View
Help
Undo
Cut
Copy
Paste
Delete
Select All
Right to left Reading order
Show Unicode control characters
Insert Unicode control character
Open IME
Reconversion

After copying the path, click the “Edit” menu and then select “Replace…”.



```
c:\  xtfile1.txt    0
c:\  xtfile2.txt    0
c:\  xtfile3.txt    0
c:\  xtfile4.txt    0
c:\  xtfile5.txt    0
c:\  xtfile6.txt    0
c:\  xtfile7.txt    0
c:\  xtfile8.txt    0
c:\  xtfile9.txt    0
c:\  xtfile10.txt   0
c:\  xtfile11.txt   0
c:\  xtfile12.txt   0
```

In the dialog box, paste (ctrl-V) the path into the “Find what:” box. Then type the path without the spaces, followed by “\” in the “replace with:” box.

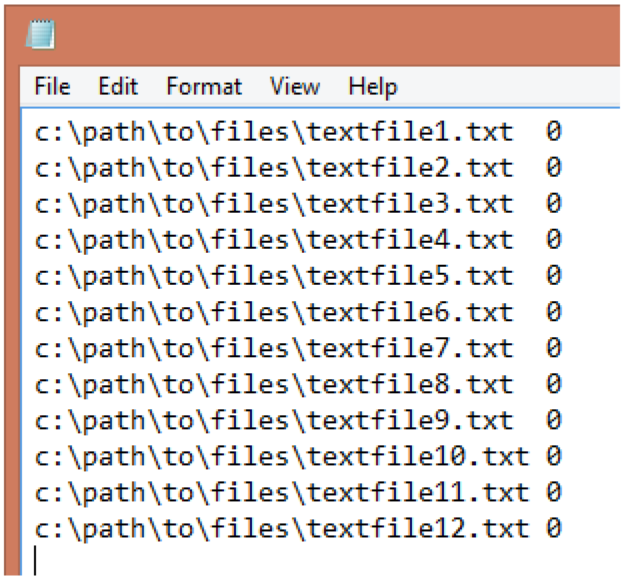


```
c:\path\to\files        textfile1.txt  0
c:\path\to\files        textfile2.txt  0
c:\path\to\files        textfile3.txt  0
c:\path\to\files        textfile4.txt  0
c:\pat
c:\pat
c:\pat
c:\pat
c:\pat
c:\pat
c:\pat
c:\pat
```

Find what: c:\path\to\files

Replace with: c:\path\to\files\

In the same dialog box, click the “Replace” button and then click cancel or close the dialog box. Your file should be in the format expected by *Udat*.



```
c:\path\to\files\textfile1.txt  0
c:\path\to\files\textfile2.txt  0
c:\path\to\files\textfile3.txt  0
c:\path\to\files\textfile4.txt  0
c:\path\to\files\textfile5.txt  0
c:\path\to\files\textfile6.txt  0
c:\path\to\files\textfile7.txt  0
c:\path\to\files\textfile8.txt  0
c:\path\to\files\textfile9.txt  0
c:\path\to\files\textfile10.txt 0
c:\path\to\files\textfile11.txt 0
c:\path\to\files\textfile12.txt 0
```

Save the file by clicking the “File” menu and then “Save As…”. The file will be the input file for *Udat*.

Important: UDAT can also analyze the correlation between a text file and a value. For an analysis of correlation, the first value needs to be a non-integer number. For instance, you can change the number 0 to 0.0, so that the first line will be: “c:\path\to\files\textfile1.text 0.0”.

## 2. Analyzing the text files

For analyzing the text files we will use a tool that can analyze the text files automatically.

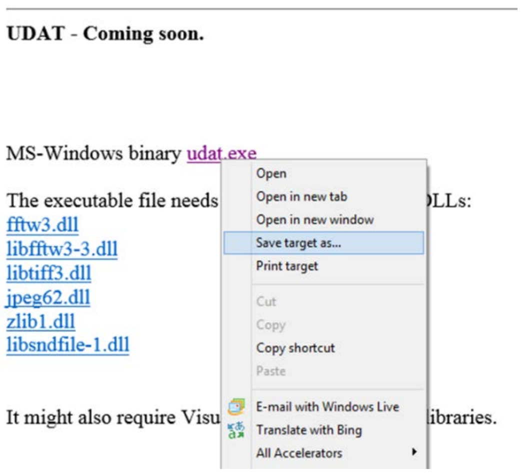
**What is UDAT?** *UDAT* (Unified Data Analysis Tool) is a command-line program that can analyze text files. It computes a large set of several hundred numerical content descriptors that reflect the text. For instance, the average word length is a simple numerical content descriptors. *UDAT* can perform classification of text files, as well as compute the correlation of text files with continuous values.

### 2.1. Downloading and running *UDAT*

To analyze the files, we will use the *UDAT* program. To perform the analysis, we will need to download *UDAT* to our computers. The files can be downloaded from

http://vfacstaff.ltu.edu/lshamir/downloads/udat/

the folder contains these files:



UDAT - Coming soon.

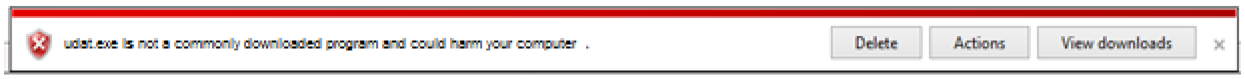
MS-Windows binary udat.exe

The executable file needs ... DLLs:
fftw3.dll
libfftw3-3.dll
libtiff3.dll
jpeg62.dll
zlib1.dll
libsndfile-1.dll

It might also require Visu... libraries.

You will need to right click the file “udat.exe” and choose “Save target as…” or “Save link as…” and save the file in your hard drive. For convenience, it is better to name the folder “udat”.

When you download the file, your browser might not recognize the file and might therefore give you a warning.



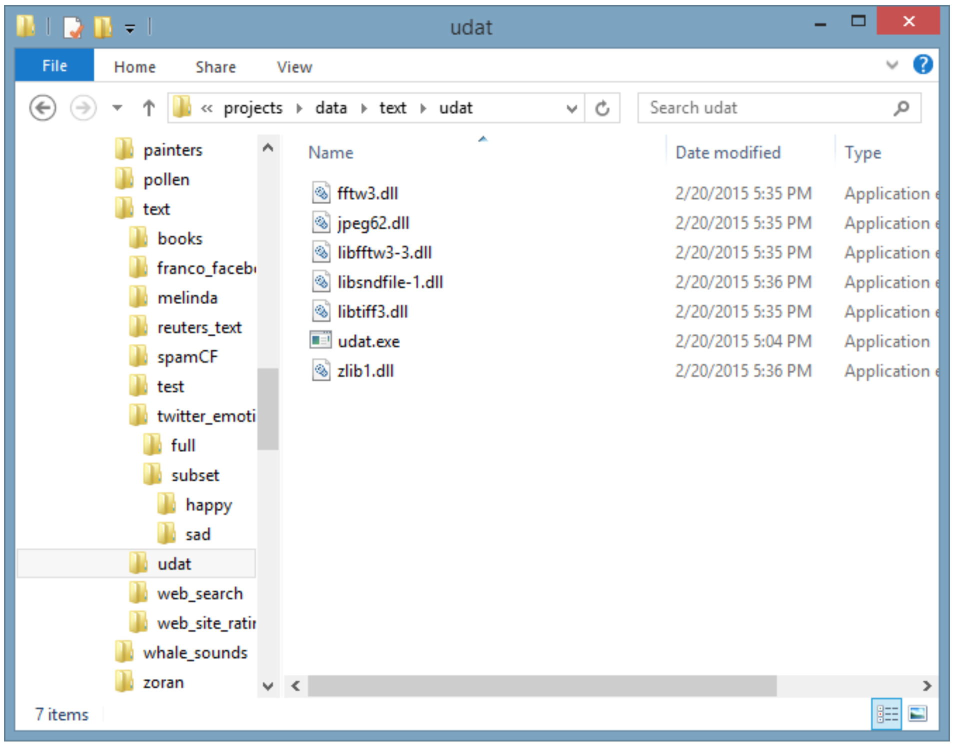
udat.exe is not a commonly downloaded program and could harm your computer .

In that case, you should click "Actions" and then select "download anyway". The file is safe to download and is certainly not malicious.

Then, you will need to right click the .dll files and save them in the same folder where you downloaded "udat.exe". The files that you need to download and save are:

udat.exe
fftw3.dll
libfftw3-3.dll
libtiff3.dll
jpeg62.dll
zlib1.dll
libsndfile-1.dll

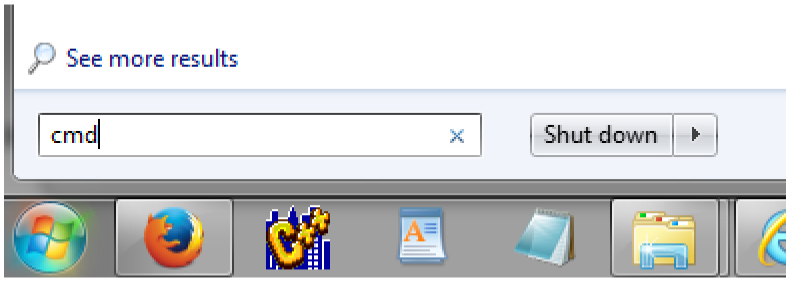
When done, your folder should contain the *Udat* files:



| Name | Date modified | Type |
|---|---|---|
| fftw3.dll | 2/20/2015 5:35 PM | Application |
| jpeg62.dll | 2/20/2015 5:35 PM | Application |
| libfftw3-3.dll | 2/20/2015 5:35 PM | Application |
| libsndfile-1.dll | 2/20/2015 5:36 PM | Application |
| libtiff3.dll | 2/20/2015 5:35 PM | Application |
| udat.exe | 2/20/2015 5:04 PM | Application |
| zlib1.dll | 2/20/2015 5:36 PM | Application |

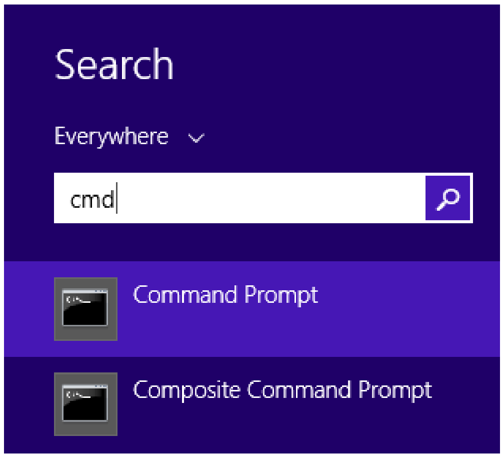
**What are DLLs?** A Dynamic Link Library (DLL) is a file that contains machine code (programs). Applications that run on the computers can use that code (link to it) to perform common tasks that the DLL program does. The advantage of the DLL is that several different applications that need to perform the same task as part of their algorithm can use the same DLL, and that saves computer resources. Accessing the programs in the DLL is done through functions that the DLL *exports* to other applications.

*UDAT* is a program that runs through the command-line. To use it, we first need to launch the command-line window. In Windows 7, that can be done by clicking the button at the left of your taskbar or pressing the right Windows key on your keyboard, and then typing "cmd" in the search box.



Then select "cmd.exe" to open the command line window.

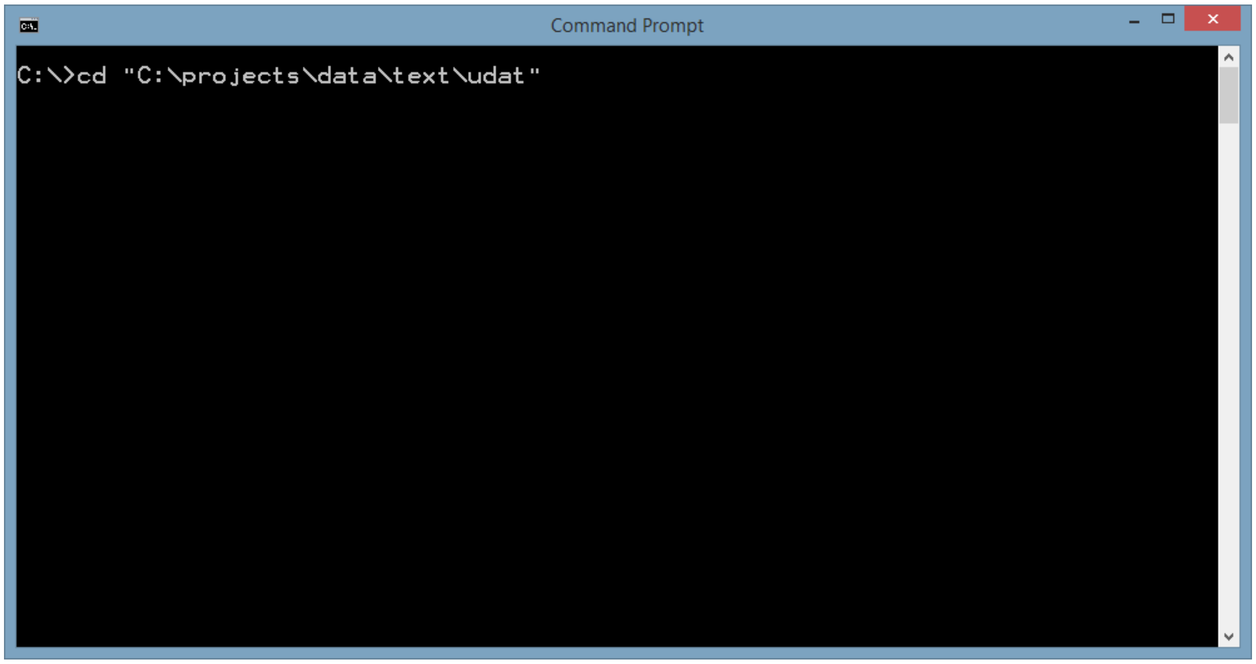
In Windows 8.1, you need to click the Windows button and then type "cmd" or "command prompt".



Then, select "Command Prompt" to open the command-line interface window.

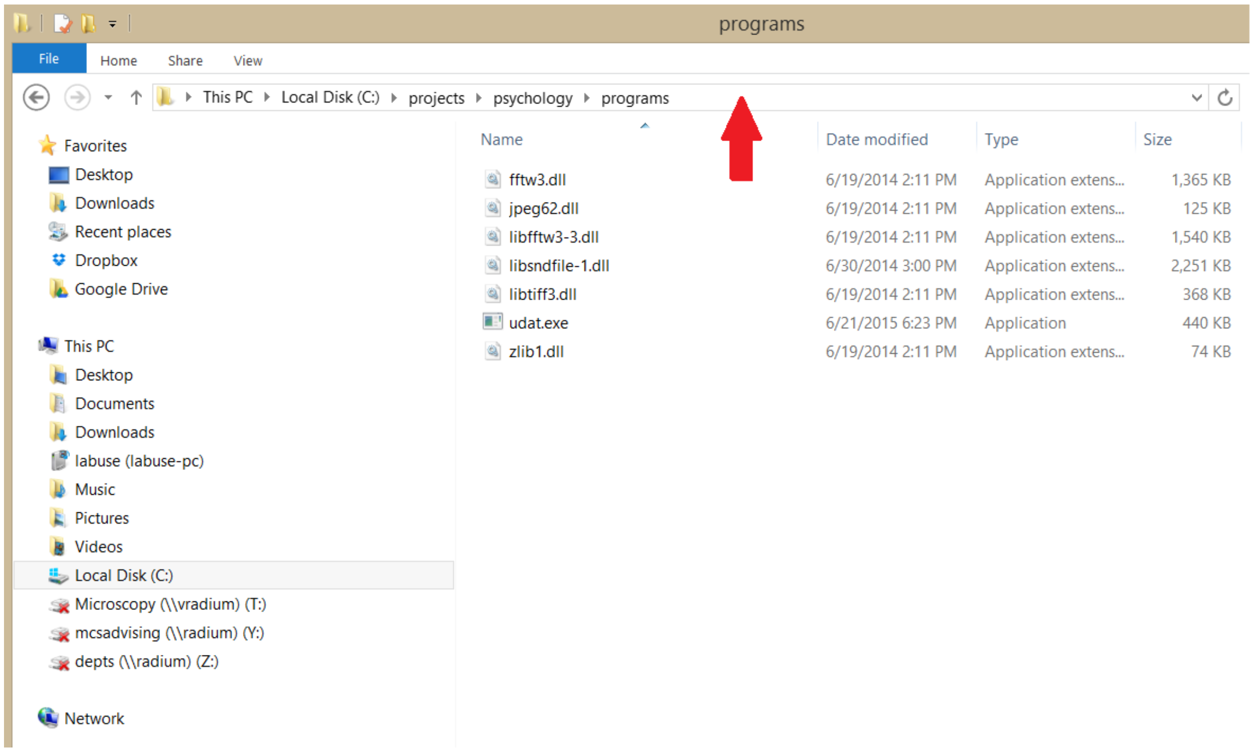
**What is a command-line interface?** Unlike graphical user interface in which the user controls the computer using a mouse or touchscreen, in a command-line interface all interactions with the computer are done through text commands. The user types commands as input, and the computer produces the output on the same window. Command-line interfaces were very common before Windows became popular, and are still used, normally in servers. Most operating systems such as Mac OS and Windows also have the option to control the computer through a command-line interface. Command-line interfaces require to memorize the commands, but on the other hand they allow running complex commands performing tasks that cannot be done by using the graphical user interface.

In the command-line window, you need to change the path to the path of the "udat" folder. By using the command "cd", followed by the path to the folder where you saved udat.exe.



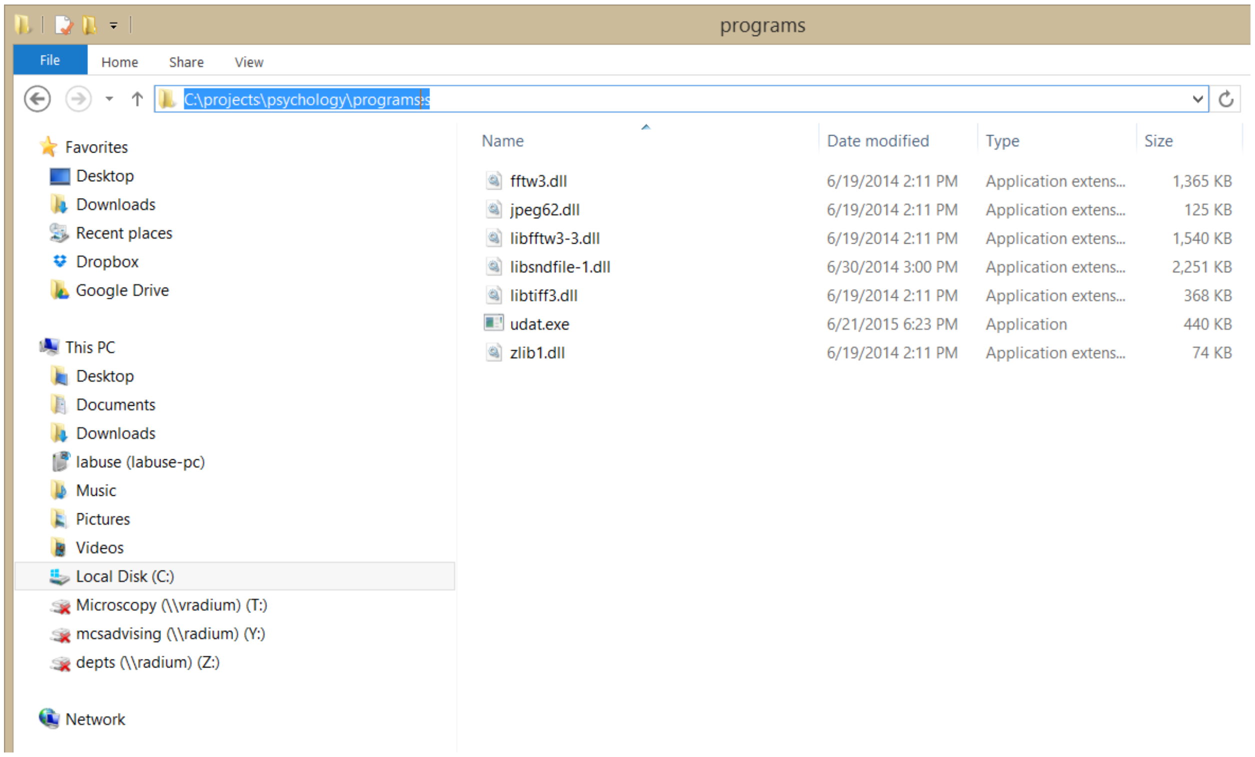
```
C:\>cd "C:\projects\data\text\udat"
```

If you do not know the full path to the folder where UDAT.exe is, you can find it with your file explorer, and then click the box near the top of the file explorer window, right of the name of the folder ("programs" in this case), as marked with the red arrow below:



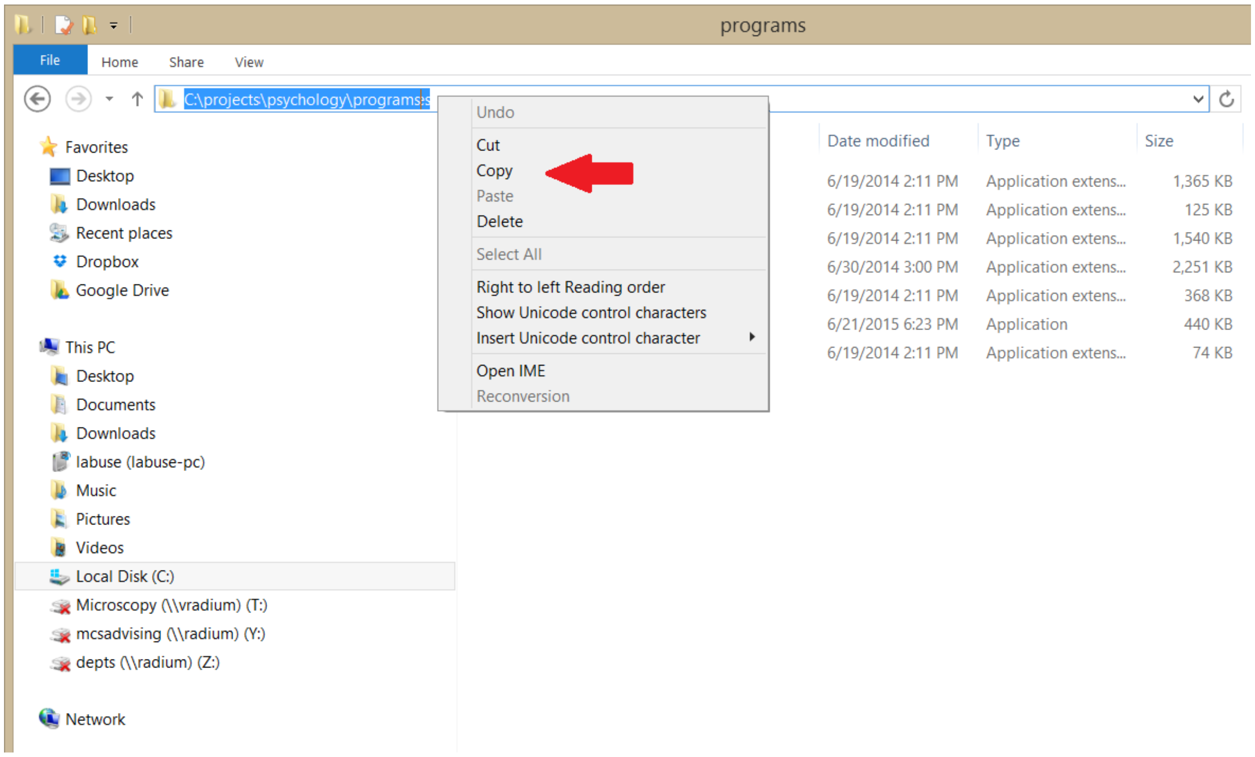
| Name | Date modified | Type | Size |
|---|---|---|---|
| fftw3.dll | 6/19/2014 2:11 PM | Application extens... | 1,365 KB |
| jpeg62.dll | 6/19/2014 2:11 PM | Application extens... | 125 KB |
| libfftw3-3.dll | 6/19/2014 2:11 PM | Application extens... | 1,540 KB |
| libsndfile-1.dll | 6/30/2014 3:00 PM | Application extens... | 2,251 KB |
| libtiff3.dll | 6/19/2014 2:11 PM | Application extens... | 368 KB |
| udat.exe | 6/21/2015 6:23 PM | Application | 440 KB |
| zlib1.dll | 6/19/2014 2:11 PM | Application extens... | 74 KB |

The box will then show the path of the folder, and the text will be highlighted.



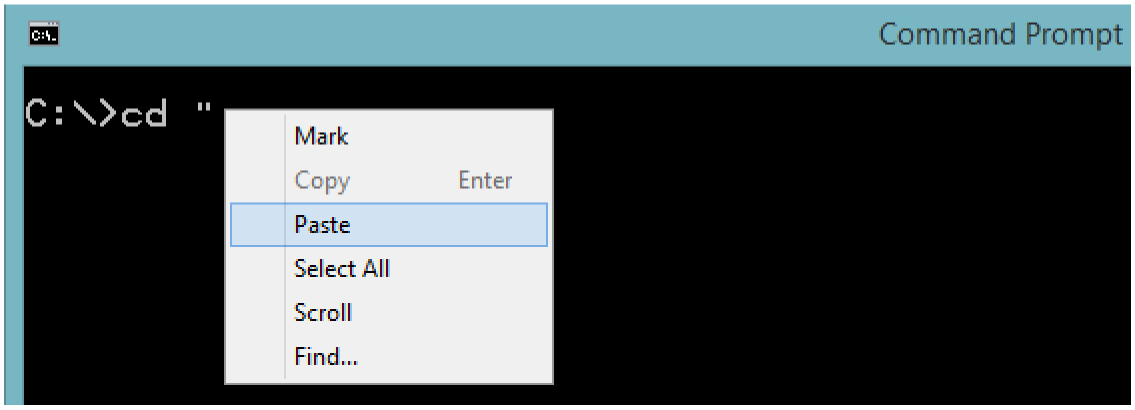
| Name | Date modified | Type | Size |
|---|---|---|---|
| fftw3.dll | 6/19/2014 2:11 PM | Application extens... | 1,365 KB |
| jpeg62.dll | 6/19/2014 2:11 PM | Application extens... | 125 KB |
| libfftw3-3.dll | 6/19/2014 2:11 PM | Application extens... | 1,540 KB |
| libsndfile-1.dll | 6/30/2014 3:00 PM | Application extens... | 2,251 KB |
| libtiff3.dll | 6/19/2014 2:11 PM | Application extens... | 368 KB |
| udat.exe | 6/21/2015 6:23 PM | Application | 440 KB |
| zlib1.dll | 6/19/2014 2:11 PM | Application extens... | 74 KB |

Then you can right-click the text box and select “copy” to copy the path to your clipboard.



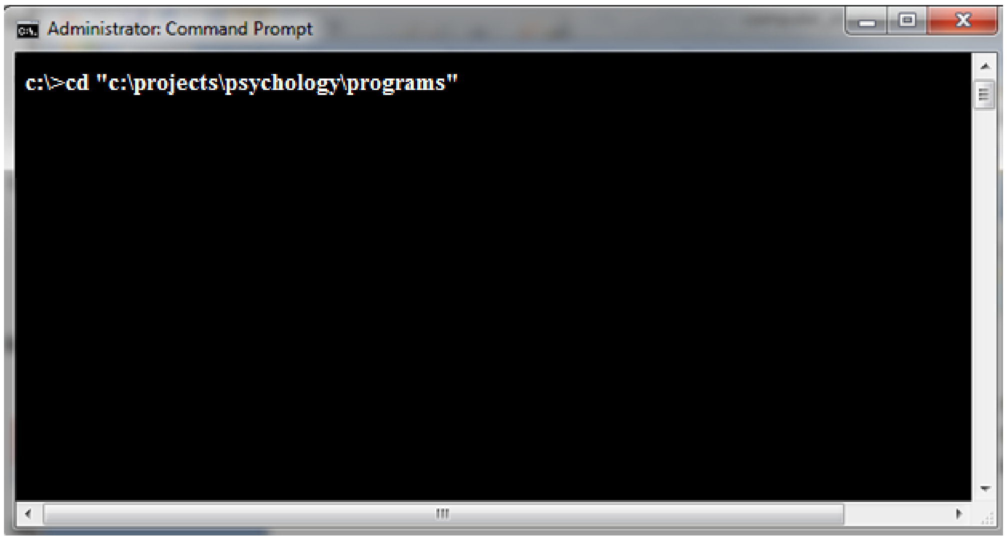
| Date modified | Type | Size |
|---|---|---|
| 6/19/2014 2:11 PM | Application extens... | 1,365 KB |
| 6/19/2014 2:11 PM | Application extens... | 125 KB |
| 6/19/2014 2:11 PM | Application extens... | 1,540 KB |
| 6/30/2014 3:00 PM | Application extens... | 2,251 KB |
| 6/19/2014 2:11 PM | Application extens... | 368 KB |
| 6/21/2015 6:23 PM | Application | 440 KB |
| 6/19/2014 2:11 PM | Application extens... | 74 KB |

And then paste it to the command line window by right-clicking the command prompt window and selecting “paste”. The path should be inside quotation marks (“).



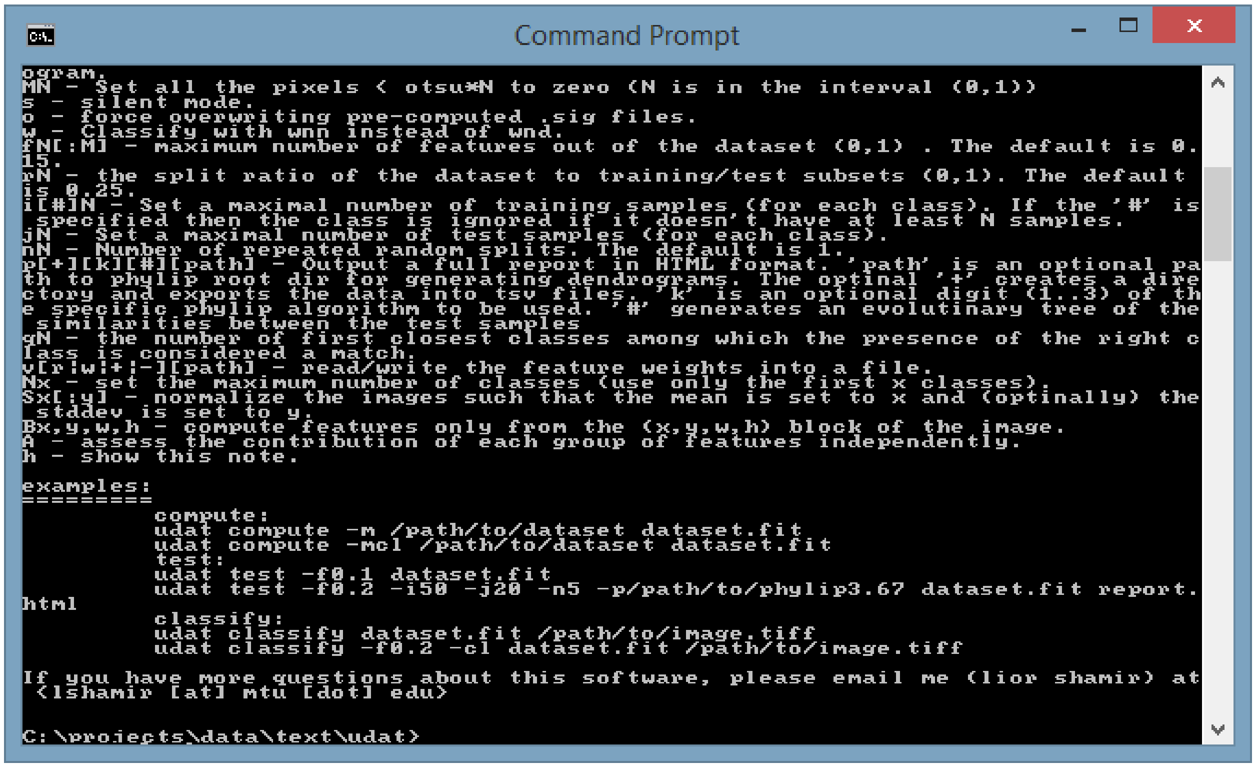
```
C:\>cd "
```

Then press <Enter>. If you copied the path manually, make sure to type the full path correctly.



```
c:\>cd "c:\projects\psychology\programs"
```

Then, type “udat” and press <Enter>. *UDAT* should display a short instruction page.



```
ogram.
MN - Set all the pixels < otsu*N to zero (N is in the interval (0,1))
s - silent mode.
o - force overwriting pre-computed .sig files.
w - Classify with wnn instead of wnd.
fN[:M] - maximum number of features out of the dataset (0,1) . The default is 0.
15.
rN - the split ratio of the dataset to training/test subsets (0,1). The default
is 0.25.
i[#]N - Set a maximal number of training samples (for each class). If the '#' is
 specified then the class is ignored if it doesn't have at least N samples.
jN - Set a maximal number of test samples (for each class).
nN - Number of repeated random splits. The default is 1.
p[+][k][#][path] - Output a full report in HTML format. 'path' is an optional pa
th to phylip root dir for generating dendrograms. The optinal '+' creates a dire
ctory and exports the data into tsv files. 'k' is an optional digit (1..3) of th
e specific phylip algorithm to be used. '#' generates an evolutinary tree of the
 similarities between the test samples
gN - the number of first closest classes among which the presence of the right c
lass is considered a match.
v[r|w|+|-][path] - read/write the feature weights into a file.
Nx - set the maximum number of classes (use only the first x classes).
Sx[:y] - normalize the images such that the mean is set to x and (optinally) the
 stddev is set to y.
Bx,y,w,h - compute features only from the (x,y,w,h) block of the image.
A - assess the contribution of each group of features independently.
h - show this note.

examples:
=========
          compute:
          udat compute -m /path/to/dataset dataset.fit
          udat compute -mcl /path/to/dataset dataset.fit
          test:
          udat test -f0.1 dataset.fit
          udat test -f0.2 -i50 -j20 -n5 -p/path/to/phylip3.67 dataset.fit report.
html
          classify:
          udat classify dataset.fit /path/to/image.tiff
          udat classify -f0.2 -cl dataset.fit /path/to/image.tiff

If you have more questions about this software, please email me (lior shamir) at
 <lshamir [at] mtu [dot] edu>

C:\projects\data\text\udat>
```

## 2.2. Computing text features

**What is a text feature?** A text feature (or numerical text content descriptor) is a number that reflects the text content. For instance, the average length of word in the document is a simple number that describes the text. Other number can be the average words in a sentence in the document, amount of words describing emotions, amount of words related to a certain topic (e.g., legal words, furniture words, etc'), use of punctuations, and more. In our experiment we use the *UDAT* tool to compute a large set of numerical text content descriptors for each post. Each feature contains a small amount of information about the post, but all features together contain substantial amount of information that reflects many different aspects of the text content that we process.

### 2.2.1 Computing text features when the files are in different folders

After *UDAT* is installed, we can use *UDAT* to convert each text file to a set of numerical descriptors. The command line that will compute the numerical descriptors for each text file is the following:

```
udat compute –m c:\data\twitter_emotions\subset c:\data\twitter_emotions\subset\output.fit
```

The path is the root path of the classes folders, and the output file “output.fit” can be created anywhere on the hard drive, but it is convenient to create it in the same folder.

After the file names will be displayed on the screen, the file “output.fit” will be created in the “c:\data\twitter_emotions\subset” folder.

### 2.2.2 Computing text features using a main file

In Section 1.2 we showed how to create a text file that contains all file paths and their corresponding values. Assuming that the name of the file is files.cor, the command line for computing the text features of the files as follows:

```
udat compute  –m c:\data\files.cor c:\data\output.fit
```

The file “c:\data\files.fit” and “c:\data\output.fit” can be created in any valid folder on the computer’s hard drive.

## 2.3. Automatic classification and analysis of the text

For the automatic analysis of the text features we will use *UDAT*'s pattern recognition capabilities.

**What is pattern recognition?** In the context of computer science, pattern recognition is the ability of the computer to find patterns in the data. In our experiment we look for patterns in the numerical values computed from each post (text file). Each class has many posts in it, and the purpose of the pattern recognition algorithm is to find automatically the numerical values that are repetitive in posts of that class, but not in the other classes. These patterns can make a distinction between one class and the other classes, and can be used to numerically characterize a certain trends in the posts of social media users.

### 2.3.1 Classification

We can test how well the computer can differentiate between the different groups. That can be done by using the "test" command of *UDAT*:

```
udat test –f0.35  –i40  –j10  –n20 -w c:\data\output.fit
```

"-f0.35" tells *UDAT* to use 35% of the numerical content descriptors.

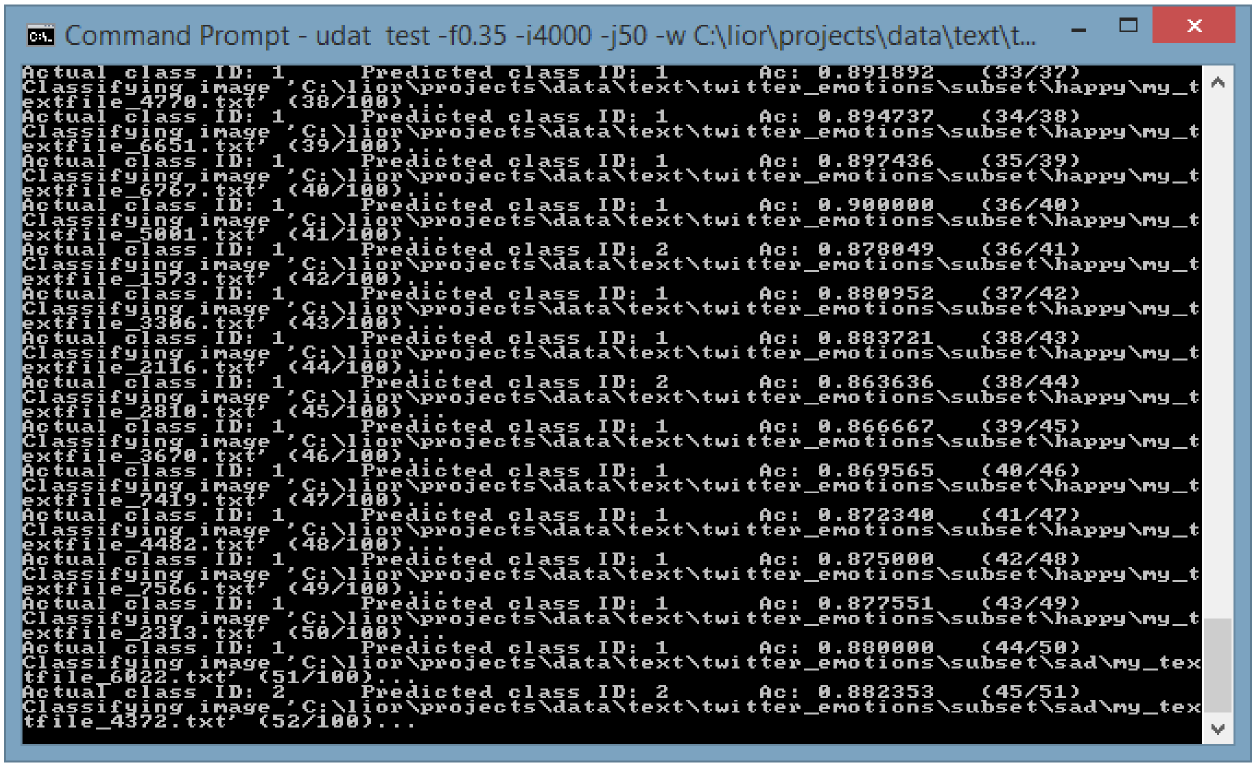
"-i45" and "-j20" means that 45 text files from each class will be used for training and 20 will be used for testing.

"-n20" tells *UDAT* to repeat the experiment 20 times, such that in each run different text files will be randomly allocated to training and test sets. That is called *cross-validation*.

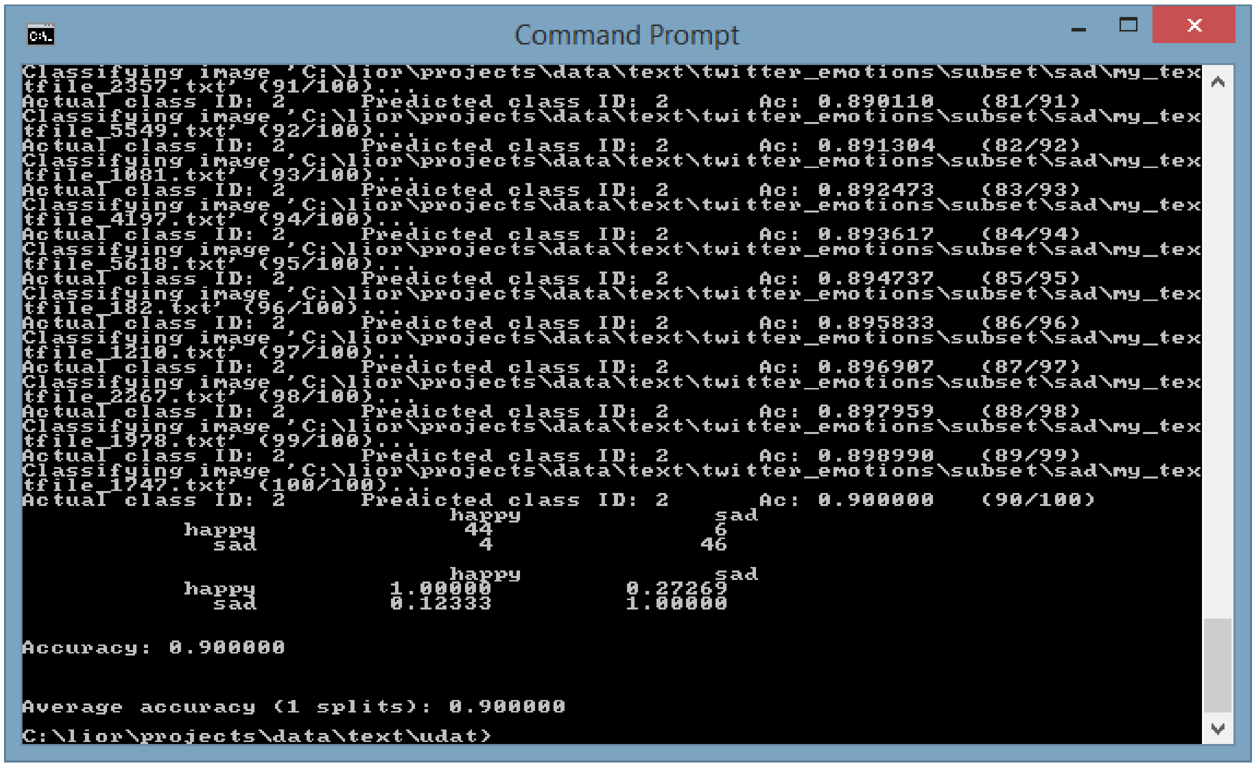
"-w" means that *UDAT* will use the Weighted Nearest Distance (WND) pattern recognition method.

**What is cross-validation?** Supervised machine learning systems are trained with some samples so that the algorithm can identify repetitive patterns in the different classes, and then the performance of the algorithm is evaluated using the rest of the samples. In some cases it leaves just few samples for testing. For instance, if just 10 samples are left for testing, and all 10 of them are classified correctly, the conclusion might be that the classifier is always correct, although the 10 samples could have all been classified correctly by chance. To increase the number of test samples, the experiment is repeated multiple times such that in each run different samples are allocated for training and testing. For instance, if we have a total of 100 samples and we use 10 samples for testing, we can run it 10 times and in each run different 10 images will be used for testing.

When running, *UDAT* will show the files that are being classified, their class ID and their predicted class ID. If the class ID and the predicted class ID are the same, the classification of that text file is correct.


```
Command Prompt - udat  test -f0.35 -i4000 -j50 -w C:\lior\projects\data\text\t...
Actual class ID: 1      Predicted class ID: 1       Ac: 0.891892    (33/37)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\happy\my_t
extfile_4770.txt' (38/100)...
Actual class ID: 1      Predicted class ID: 1       Ac: 0.894737    (34/38)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\happy\my_t
extfile_6651.txt' (39/100)...
Actual class ID: 1      Predicted class ID: 1       Ac: 0.897436    (35/39)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\happy\my_t
extfile_6767.txt' (40/100)...
Actual class ID: 1      Predicted class ID: 1       Ac: 0.900000    (36/40)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\happy\my_t
extfile_5001.txt' (41/100)...
Actual class ID: 1      Predicted class ID: 2       Ac: 0.878049    (36/41)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\happy\my_t
extfile_1573.txt' (42/100)...
Actual class ID: 1      Predicted class ID: 1       Ac: 0.880952    (37/42)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\happy\my_t
extfile_3306.txt' (43/100)...
Actual class ID: 1      Predicted class ID: 1       Ac: 0.883721    (38/43)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\happy\my_t
extfile_2116.txt' (44/100)...
Actual class ID: 1      Predicted class ID: 2       Ac: 0.863636    (38/44)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\happy\my_t
extfile_2810.txt' (45/100)...
Actual class ID: 1      Predicted class ID: 1       Ac: 0.866667    (39/45)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\happy\my_t
extfile_3670.txt' (46/100)...
Actual class ID: 1      Predicted class ID: 1       Ac: 0.869565    (40/46)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\happy\my_t
extfile_7419.txt' (47/100)...
Actual class ID: 1      Predicted class ID: 1       Ac: 0.872340    (41/47)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\happy\my_t
extfile_4482.txt' (48/100)...
Actual class ID: 1      Predicted class ID: 1       Ac: 0.875000    (42/48)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\happy\my_t
extfile_7566.txt' (49/100)...
Actual class ID: 1      Predicted class ID: 1       Ac: 0.877551    (43/49)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\happy\my_t
extfile_2313.txt' (50/100)...
Actual class ID: 1      Predicted class ID: 1       Ac: 0.880000    (44/50)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\sad\my_tex
tfile_6022.txt' (51/100)...
Actual class ID: 2      Predicted class ID: 2       Ac: 0.882353    (45/51)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\sad\my_tex
tfile_4372.txt' (52/100)...
```

*UDAT* will test 20 text files per class in each of the 20 runs, and when done will show the average classification accuracy of the text files. When done, it will show the classification accuracy.



```
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\sad\my_tex
tfile_2357.txt' (91/100)...
Actual class ID: 2      Predicted class ID: 2       Ac: 0.890110    (81/91)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\sad\my_tex
tfile_5549.txt' (92/100)...
Actual class ID: 2      Predicted class ID: 2       Ac: 0.891304    (82/92)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\sad\my_tex
tfile_1081.txt' (93/100)...
Actual class ID: 2      Predicted class ID: 2       Ac: 0.892473    (83/93)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\sad\my_tex
tfile_4197.txt' (94/100)...
Actual class ID: 2      Predicted class ID: 2       Ac: 0.893617    (84/94)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\sad\my_tex
tfile_5618.txt' (95/100)...
Actual class ID: 2      Predicted class ID: 2       Ac: 0.894737    (85/95)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\sad\my_tex
tfile_182.txt' (96/100)...
Actual class ID: 2      Predicted class ID: 2       Ac: 0.895833    (86/96)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\sad\my_tex
tfile_1210.txt' (97/100)...
Actual class ID: 2      Predicted class ID: 2       Ac: 0.896907    (87/97)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\sad\my_tex
tfile_2267.txt' (98/100)...
Actual class ID: 2      Predicted class ID: 2       Ac: 0.897959    (88/98)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\sad\my_tex
tfile_1978.txt' (99/100)...
Actual class ID: 2      Predicted class ID: 2       Ac: 0.898990    (89/99)
Classifying image 'C:\lior\projects\data\text\twitter_emotions\subset\sad\my_tex
tfile_1747.txt' (100/100)...
Actual class ID: 2      Predicted class ID: 2       Ac: 0.900000    (90/100)
                                 happy              sad
           happy                  44                 6
             sad                   4                46

                                 happy              sad
           happy           1.00000          0.27269
             sad           0.12333          1.00000


Accuracy: 0.900000



Average accuracy (1 splits): 0.900000

C:\lior\projects\data\text\udat>
```

In the case above, *UDAT* was able to associate a text file with the class with average accuracy of 90.00%. Any classification accuracy that is higher than random guessing is an indication that the machine learning system is informative in identifying paintings by their style. For instance, if we have two classes, classification accuracy higher than 50% indicates that the system is informative in the association of the paintings.

**What is classification accuracy?** The most basic way of measuring the performance of pattern recognition systems is the classification accuracy. From each class of images we allocate some posts (text files) to training the classifier, so that the pattern recognition algorithm can identify repetitive patterns in the values of the text features computed from these posts. The rest of the posts are used for testing. Each test post is classified, and the prediction of the classifier is compared to the actual class that the post belongs in. If the predicted class is the same as the actual class the classification is considered a hit, and if the two are different the classification is a miss. After classifying all test posts, the classification accuracy is determined by the number of hits divided by the total number of classifications (the total number of test posts).

Now, we can change the number of features that we use by changing the value of the "-f" switch. For instance, to use 55% of the numerical image content descriptors we can use the following command line:

```
udat test  –f0.55  –i40  –j20  –n20 -w c:\data\output.fit
```

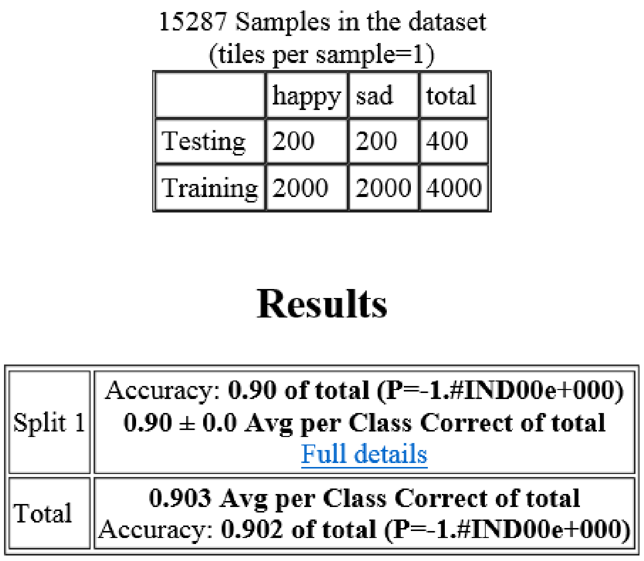
We will need to run the same command with “-f0.05”, “-f0.1”, “-f0.15”, “-f0.3”, “-f0.4”, “-f0.5”. The amount of content descriptors that provides the highest classification accuracy will be the one used in our following experiments.

We can also ask *UDAT* to create a detailed report file showing the results of the experiment. That will be done by using the “-p” switch, and specifying another file in the end of the command line:

```
udat test –f0.55  –i40  –j20  –n20  -w -p  c:\data\output.fit  c:\data\results.html
```

The file “results.html” will provide a detailed report on the experiment. To open the report file, just double click it and it will be open in your default browser. The file will open in your default browser.

Here is an example of a report file of an experiment with:



15287 Samples in the dataset
(tiles per sample=1)

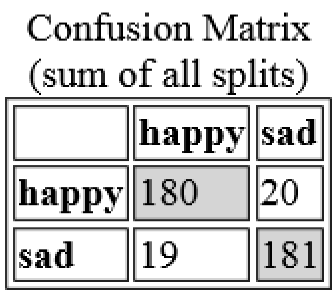
| | happy | sad | total |
|---|---|---|---|
| Testing | 200 | 200 | 400 |
| Training | 2000 | 2000 | 4000 |

**Results**

| | |
|---|---|
| Split 1 | Accuracy: **0.90 of total (P=-1.#IND00e+000)** **0.90 ± 0.0 Avg per Class Correct of total** Full details |
| Total | **0.903 Avg per Class Correct of total** Accuracy: **0.902 of total (P=-1.#IND00e+000)** |

The upper part of the form shows the classes that were used in the experiment. In the specific report there are two classes such that each class contains twitter posts of a certain emotion – happy or sad. The report specifies the names of the classes (in this cases the emotions), and the number of training and test images for each class. According to the report, the classification accuracy of a painting to a painter is ~0.90, or 90%.

Below the summary of the classification accuracy you will find the confusion matrix. The confusion matrix reflects how well the computer analysis can associate text files to their classes.

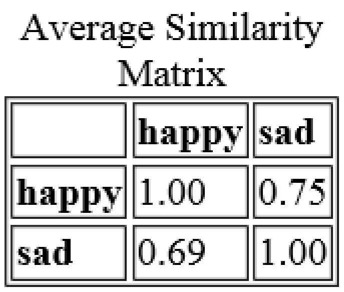

Confusion Matrix
(sum of all splits)

| | happy | sad |
|---|---|---|
| **happy** | 180 | 20 |
| **sad** | 19 | 181 |

For instance, the confusion matrix shows that out of 200 happy posts, 20 were incorrectly classified as sad. The confusion matrix therefore reflects the confusion of the algorithm when classifying the text files.

**What is a confusion matrix?** A confusion matrix is a visualization of the results of a supervised machine learning experiment. The rows of the matrix represent the actual classes that the sample belong in, and the columns represent the predicted classes as determined by the pattern recognition algorithm. The values in the cells are the numbers of the posts (text files) from each class (the rows) classified to each of the other classes (the columns). If the classifier is always accurate, all values will be on the diagonal. For instance, in the example confusion matrix above the 180 of the 200 happy posts were (correctly) classified as happy, and 20 were (incorrectly) classified as sad.

The similarity matrix shows the similarities between the different emotions as computed by the algorithm.  The similarity values are between 0 and 1, such that 1 means full similarity.


Average Similarity
Matrix

| | happy | sad |
|---|---|---|
| **happy** | 1.00 | 0.75 |
| **sad** | 0.69 | 1.00 |

**What is a similarity matrix?** The similarity matrix visualizes the differences between the classes as deduced by the pattern recognition algorithm. The cell i,j shows the similarity between class I and class j. The values are normally in the range of [0,1]. The diagonal shows the similarity of the class to itself, which is normalized to 1.

## 2.3.2. Correlation

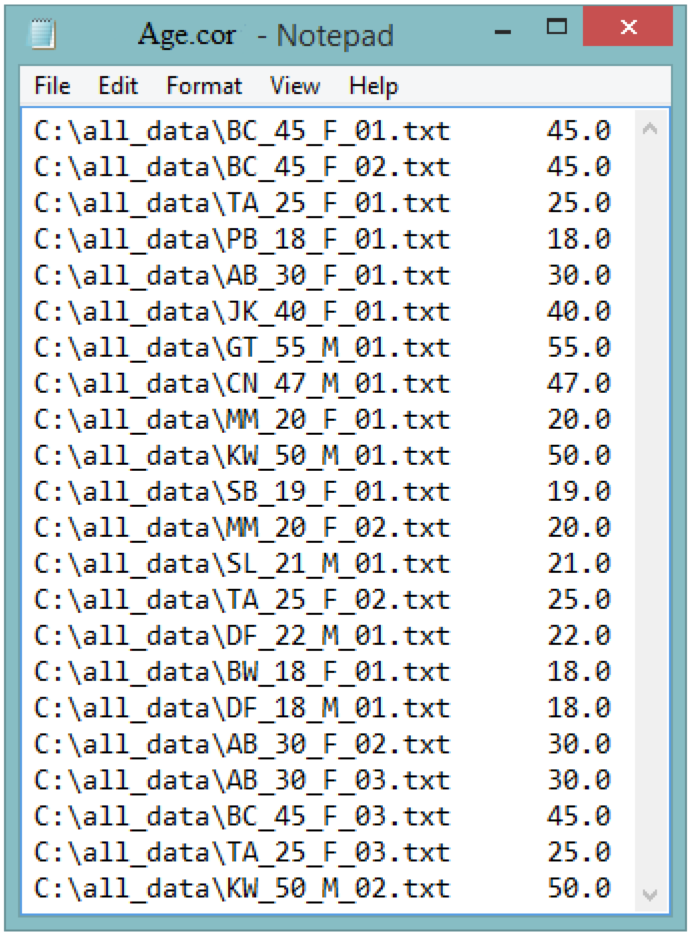
In the experiment above we used the computer analysis to assign text files of social media posts into one of two emotional states: happy and sad. The task of automatically assigning samples into one of a set of distinct classes is called *classification*. Automatic classification is an important task, but machine learning can perform more tasks. One of the useful tasks that computers can do is automatically determining the statistical correlation between text files and continuous values.

**What is statistical correlation?** Statistical correlation measures the degree of dependence between two variables. A high dependence means that one variables tends to go up when the second variables goes up, and when one of them goes down the other also tends to go down. The most common method of measuring correlation between variables is the Pearson correlation coefficient. The Pearson correlation coefficient can be any value between -1 and 1. When the correlation between the two variables is full (when one variable changes the other variable changes in the same direction) the Pearson correlation coefficient is 1. When the Pearson correlation is -1 it means inverse correlation, and 0 means that the two variables have no correlation between them.

*UDAT* can determine the correlation between the text files and the numerical values associated with them. For instance, if in the example above of happy and said posts we had not just information regarding the general emotion, but also the degree of happiness, we could identify the correlation between the text and the degree of happiness.

For instance, suppose that we want to correlate the text files with the age of the user to determine whether people at different ages express themselves differently through their social media. The following text file age.cor can be prepared in the same way the files in Section 1 were prepared, but the numbers are not discrete but continuous values (age).

**Important: The values must have a decimal point.**



```
C:\all_data\BC_45_F_01.txt      45.0
C:\all_data\BC_45_F_02.txt      45.0
C:\all_data\TA_25_F_01.txt      25.0
C:\all_data\PB_18_F_01.txt      18.0
C:\all_data\AB_30_F_01.txt      30.0
C:\all_data\JK_40_F_01.txt      40.0
C:\all_data\GT_55_M_01.txt      55.0
C:\all_data\CN_47_M_01.txt      47.0
C:\all_data\MM_20_F_01.txt      20.0
C:\all_data\KW_50_M_01.txt      50.0
C:\all_data\SB_19_F_01.txt      19.0
C:\all_data\MM_20_F_02.txt      20.0
C:\all_data\SL_21_M_01.txt      21.0
C:\all_data\TA_25_F_02.txt      25.0
C:\all_data\DF_22_M_01.txt      22.0
C:\all_data\BW_18_F_01.txt      18.0
C:\all_data\DF_18_M_01.txt      18.0
C:\all_data\AB_30_F_02.txt      30.0
C:\all_data\AB_30_F_03.txt      30.0
C:\all_data\BC_45_F_03.txt      45.0
C:\all_data\TA_25_F_03.txt      25.0
C:\all_data\KW_50_M_02.txt      50.0
```

Now we can compute the correlation between the text files and the age of the person. For that, we first need to generate the .fit file.

```
udat compute  –m c:\all_data\age.cor c:\all_data\output.fit
```

That will create the file “output.fit”. The correlation can then be determined by the following command.

```
udat test –f0.55  –i40  –j20  –n20  c:\data\output.fit
```

The arguments “-i40” and “-j20” are determined based on the number of text files that we have. In this case we assume that there are 60 text files, so 40 are allocated for training and 20 for testing. But if we had, for example, 100 files, we could use 80 for training and 20 for testing, so that the command would be “udat test –f0.55  –i80  –j20  –n20  c:\data\output.fit”.

Using a higher number of training samples normally improves the performance of machine learning algorithms.