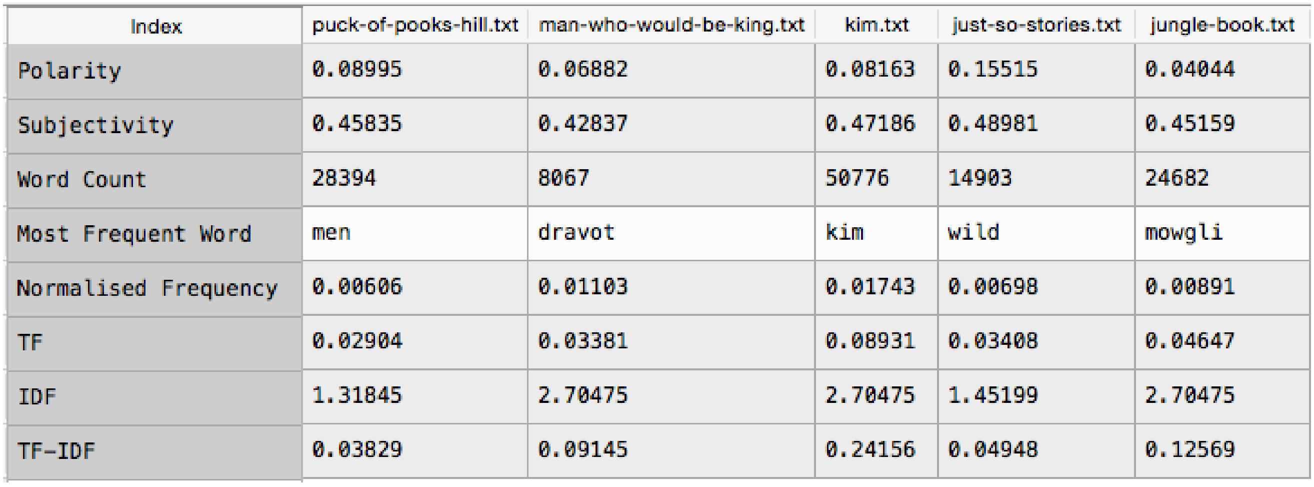
7CCSMDM1 (Data Mining)

Coursework 2

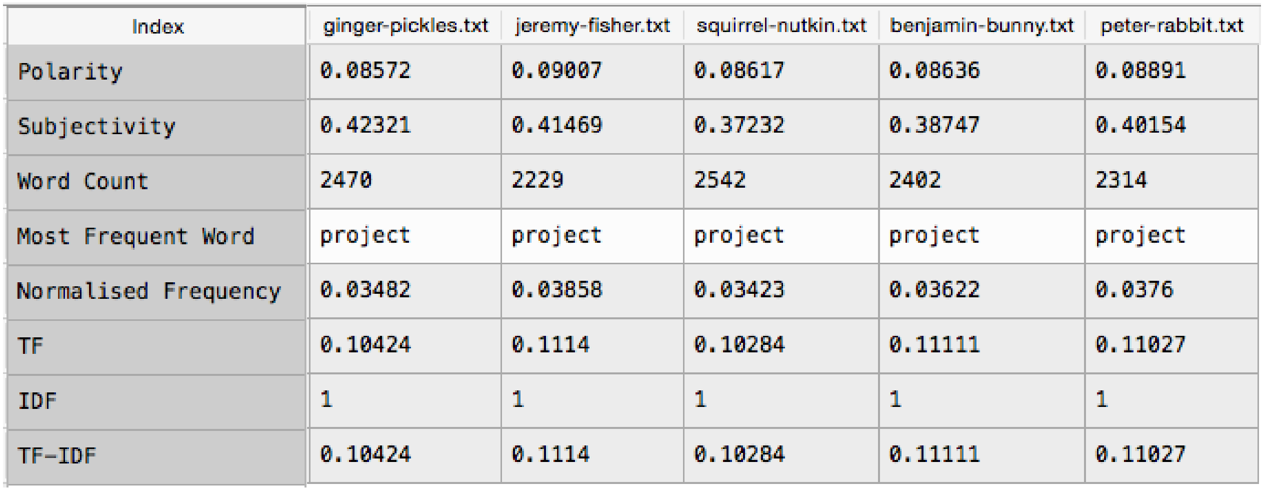
due 24th April 2020 @ 4pm GMT

1 WORKING WITH A TEXT CORPUS – NLP

<Table for all statistics>



Index	puck-of-pooks-hill.txt	man-who-would-be-king.txt	kim.txt	just-so-stories.txt	jungle-book.txt
Polarity	0.08995	0.06882	0.08163	0.15515	0.04044
Subjectivity	0.45835	0.42837	0.47186	0.48981	0.45159
Word Count	28394	8067	50776	14903	24682
Most Frequent Word	men	dravot	kim	wild	mowgli
Normalised Frequency	0.00606	0.01103	0.01743	0.00698	0.00891
TF	0.02904	0.03381	0.08931	0.03408	0.04647
IDF	1.31845	2.70475	2.70475	1.45199	2.70475
TF-IDF	0.03829	0.09145	0.24156	0.04948	0.12569



Index	ginger-pickles.txt	jeremy-fisher.txt	squirrel-nutkin.txt	benjamin-bunny.txt	peter-rabbit.txt
Polarity	0.08572	0.09007	0.08617	0.08636	0.08891
Subjectivity	0.42321	0.41469	0.37232	0.38747	0.40154
Word Count	2470	2229	2542	2402	2314
Most Frequent Word	project	project	project	project	project
Normalised Frequency	0.03482	0.03858	0.03423	0.03622	0.0376
TF	0.10424	0.1114	0.10284	0.11111	0.11027
IDF	1	1	1	1	1
TF-IDF	0.10424	0.1114	0.10284	0.11111	0.11027

<TF-IDF function>

Package: import Pipeline from sklearn.pipeline

Function: IDF(T) $=\log(\frac{1+M}{1+DF(T)})+1$

<Classifier>

2 WORKING WITH AN IMAGE DATA SET – IMAGE PROCESSING

<Explanation>

(0) pre-work

Code

```
path_list=['plantA_39.jpg','plantB_26.jpg','plantC_15.jpg',\
          'plantA_23.jpg','plantB_51.jpg','plantC_44.jpg']
for i in path_list:
    im = imageio.imread(i)
    img = color.rgb2gray(im)
```

Explanation

“path_list” is used to for-loop to read each image.

This part of for-loop is used for reading image, and use rgb2gray in order to make image become grayscale and 2-dimension array.

(a) greyscale

Code

```
    plt.imshow(img, cmap=plt.cm.gray, interpolation='nearest')
    plt.axis('off')
    plt.savefig('(a)%s_grayscale.png'%i)
```

Explanation

Produce greyscale image by ‘imshow’ command. And save images by ‘savefig’ command.

*Note: by ‘%s’ %i could make each file names different

(b) black-and-white

Code

```
    threshold = filters.threshold_otsu(img)
    binary_img = img > threshold
    plt.imshow(binary_img, cmap=plt.cm.gray, interpolation='nearest')
    plt.axis('off')
    plt.savefig('(b)%s_b&w.png'%i)
```

Explanation

Use ‘filters.threshold_otsu’to return a threshold value, and distinguish array of image become True & False.

*Note: True would show white color, False would show black color.

(c) edges

Code

```
   edges = feature.canny(img, sigma=1)


    plt.imshow(edges, cmap=plt.cm.gray, interpolation='nearest')
    plt.axis('off')
```

```
plt.savefig('(c)%s_edges.png'%i)
```

Explanation

‘feature.canny()’ is edge filter, it returns 2 values—True and False. *Note: False means empty space.

(d) contours

Code

```
threshold = filters.threshold_otsu(img)
contours = measure.find_contours(img, threshold)

for n, contour in enumerate(contours):
    plt.plot(contour[:,1], contour[:,0], 'k-', linewidth=1)
plt.axis('off')
plt.savefig('(d)%s_contours.png'%i)
```

Explanation

By using ‘measure.find.contours()’ to build an ordered collection with optimized access from its endpoints.

(e) green

Code

Explanation

(f) straight lines

Code

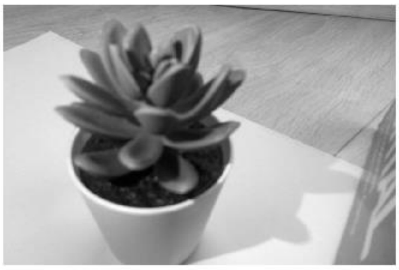
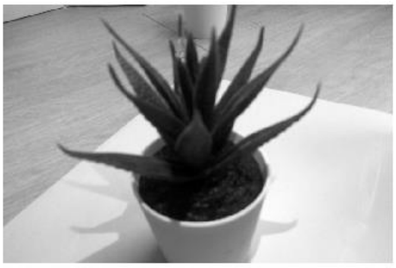
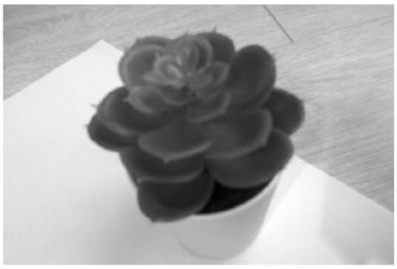
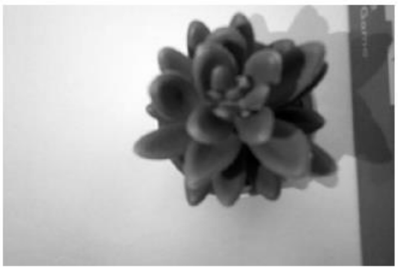
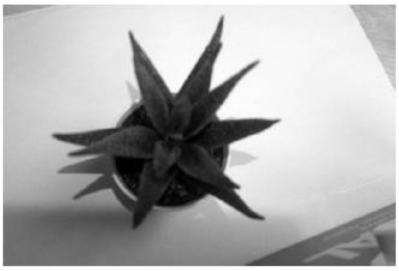
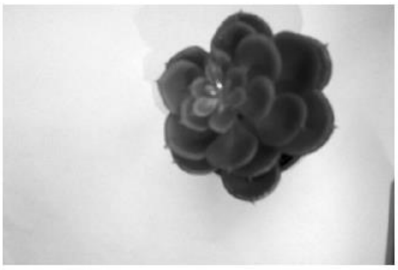
```
edges = feature.canny(img, sigma=1)
lines = transform.probabilistic_hough_line(edges,threshold=30,\
                                                       line_length=20,line_gap=3)
fig,ax0 = plt.subplots(nrows=1, ncols=1,figsize=(8, 3),sharex=True,sharey=True)

for line in lines:
    p0, p1 = line
    ax0.plot((p0[0], p1[0]),(p0[1], p1[1]))
ax0.set_xlim((0,img.shape[1]))
ax0.set_ylim((img.shape[0],0))
plt.savefig('(f)%s_straight_line.png'%i)
```

Explanation

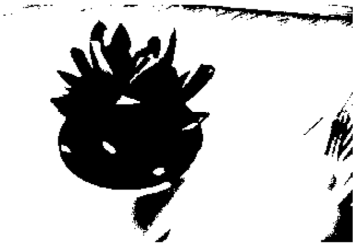
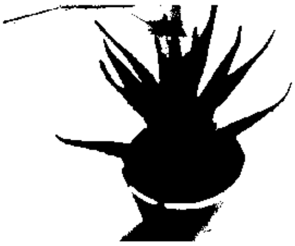
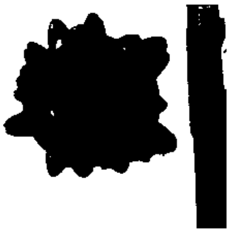
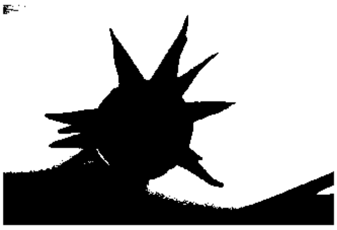
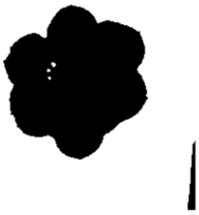
‘transform.probabilistic_hough_line()’ could create line by detecting edges. The output of this function is a list of sets.

<Result>

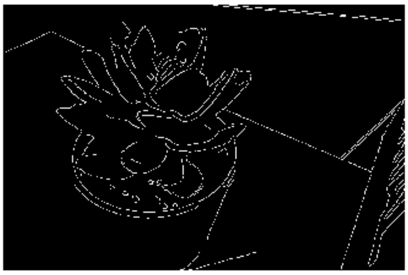
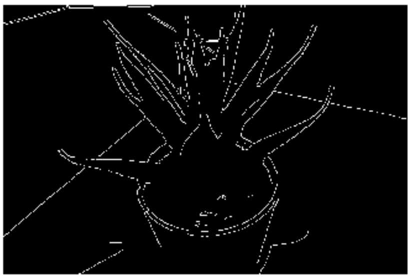
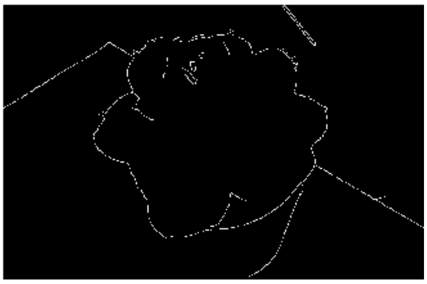
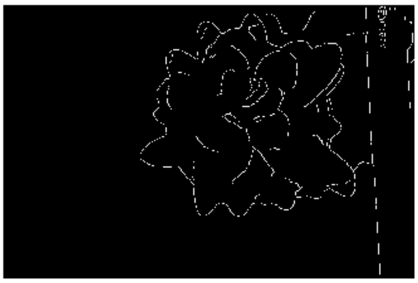
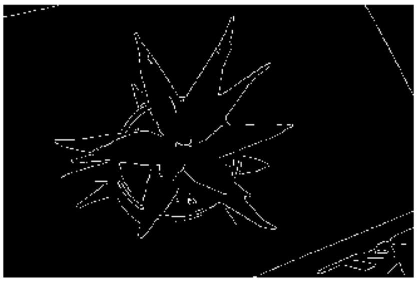
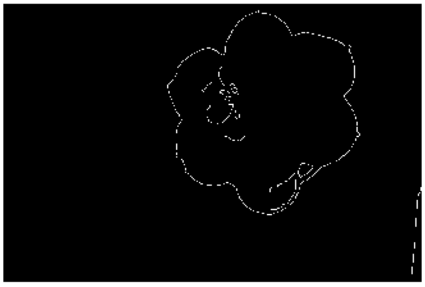
(a) Generate a greyscale version of each image.

plantA_39.jpg	plantB_26.jpg	plantC_15.jpg
		
plantA_23.jpg	plantB_51.jpg	plantC_44.jpg
		

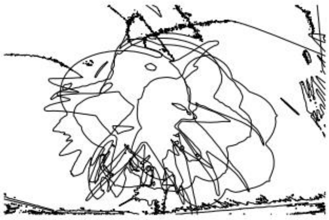
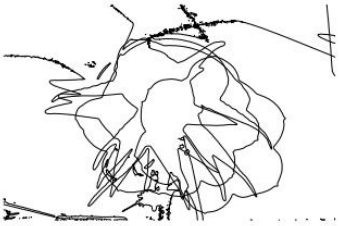
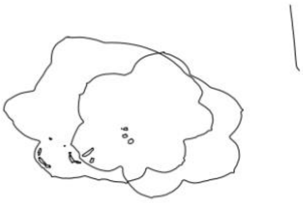
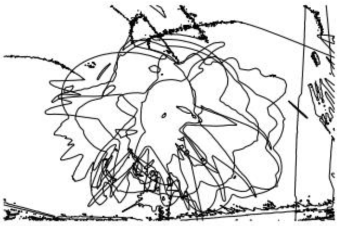
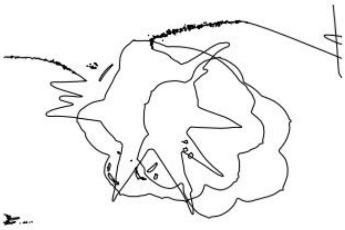
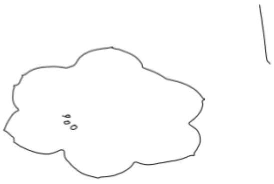
(b) Generate a black-and-white version of each image.

plantA_39.jpg	plantB_26.jpg	plantC_15.jpg
		
plantA_23.jpg	plantB_51.jpg	plantC_44.jpg
		

(c) Detect edges in each image.

plantA_39.jpg	plantB_26.jpg	plantC_15.jpg
		
plantA_23.jpg	plantB_51.jpg	plantC_44.jpg
		

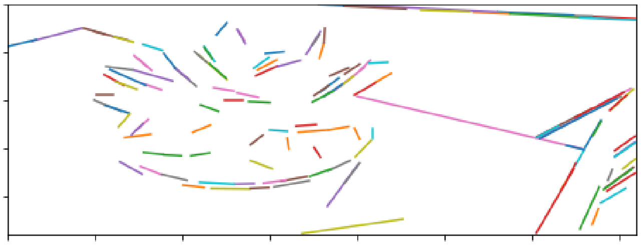
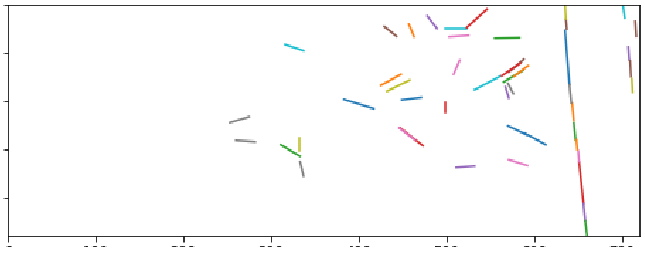
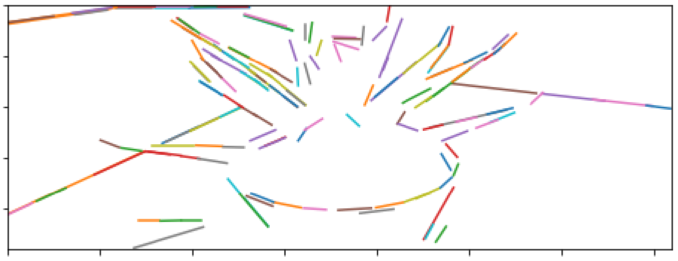
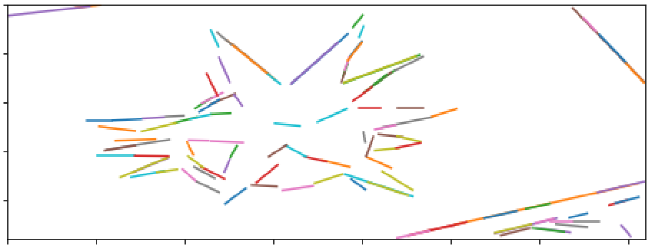
(d) Detect contours in each image.

plantA_39.jpg	plantB_26.jpg	plantC_15.jpg
		
plantA_23.jpg	plantB_51.jpg	plantC_44.jpg
		

(e) Detect the green in each image

plantA_39.jpg	plantB_26.jpg	plantC_15.jpg
plantA_23.jpg	plantB_51.jpg	plantC_44.jpg

(f) Detect straight lines in each image using the Hough transform.

plantA_39.jpg	plantA_23.jpg
	
plantB_26.jpg	plantB_51.jpg
	
plantC_15.jpg	plantC_44.jpg