Implementation of a basic CBIR using Lucene

Information Retrieval

Ignacio Amaya.
Hugo Santana.

Lucene

What is Lucene?

- Text based Search Engine
  - Spellchecking
  - Hit Highlighting
  - Analysis and tokenization
- Open Source
- Licensed by Apache

Where is it used?

OCR - Optical Character Recognition

Developed using *Python* with the library *pyocr*.

Main method used: *image_to_string*

Parameters:

- Image
- Language (From Tesseract Options)
- Builder
  - TextBuilder
  - CharBoxBuilder

EXIF - Exchangeable Image File

EXIF Contents

- Date and time information.
- Camera Information.
- Description.
- Copyright Information.

Developed using *Python* with the library *exifread*.

Main method used: *processfile*.

Sample Input for OCR and EXIF detection

Fish & Shellfish Immunology (2001) **11**, 281–291
doi:10.1006/fsim.2000.0309
Available online at http://www.idealibrary.com on IDEAL®

Immunological parameters of Javanese carp *Puntius gonionotus* (Bleeker) exposed to copper and challenged with *Aeromonas hydrophila*

M. Sharifi[1], P. A. H. L. Jayawardena[1], F. M. Yusoff[2] and R. Subasinghe[3]

[1]*Aquatic Animal Health Unit, Faculty of Veterinary Medicine,* [2]*Department of Biology, Faculty of Science, Universiti Putra Malaysia, 43400 UPM, Serdang, Selangor D. E., Malaysia, and* [3]*Fishery Resources Officer, Fisheries Department, FAO of the UN, Viale delle Terme di Caracalla, 00100 Rome, Italy*

(*Received 30 March 2000, accepted after revision 4 September 2000*)

This study was to determine the median lethal concentration (LC_{50}) of copper to Javenese carp, *Puntius gonionotus* (Bleeker), and the immune response after the fish were exposed to sublethal levels of copper and challenged with formalin killed *Aeromonas hydrophila*. The LC_{50} of copper on *P. gonionotus* at 24, 48, 72, 96 and 120 h were estimated as 2·17, 0·91, 0·57, 0·53 and 0·42 mg l^{-1}, respectively. To determine the effect of copper on the immune system, fish were exposed for 66 days to 0·05, 0·10 and 0·15 mg Cu l^{-1}. After 56 days of initial exposure to copper, fish were challenged with 0·1 ml of $4{\cdot}5 \times 10^7$ cfu ml^{-1} formalin killed *A. hydrophila* and maintained in the same concentration of copper. After the challenge, the immune response was monitored for 2 weeks using haematological and serological assays. During the initial phase of exposure to copper, significant changes were noted in the white blood cell, lysozyme, potential killing activity, total plasma protein, total immunoglobulin and haematocrit levels between the control and treated fish. One week after challenge with *A. hydrophila*, there was a significant increase in the values of white blood cells, total protein and total immunoglobulin compared to the values before the challenge. However, these values were not significantly different ($P>0{\cdot}05$) between the control and the treated fish. In contrast, NBT and lysozyme assays exhibited a significant difference ($P<0{\cdot}05$) in fish exposed to 0·10 mg Cu l^{-1} ($0{\cdot}525 \pm 0{\cdot}17$; $24{\cdot}42 \pm 3{\cdot}45 \times 10^2$ μmg ml^{-1}) and 0·15 mg Cu l^{-1} ($0{\cdot}536 \pm 0{\cdot}19$; $21{\cdot}78 \pm 1{\cdot}29 \times 10^2$ μmg ml^{-1}) compared to the control ($0{\cdot}746 \pm 0{\cdot}31$; $30{\cdot}73 \pm 5{\cdot}42 \times 10^2$ μmg ml^{-1}) after the bacterial challenge (day 61). There was however no significant difference ($P>0{\cdot}05$) in NBT and lysozyme levels in fish exposed to lower level of copper (0·05 mg Cu l^{-1}), suggesting the absence of immunosuppressive effects at lower level of exposure.

Key words: copper, sublethal effects, immunoassays, challenge test, *Puntius gonionotus*, *Aeromonas hydrophila*.

I. Introduction

Metals are released at sublethal concentration into aquatic systems from many sources. Copper at sublethal levels have been recognised as a stressor

1050–4648/01/040281 + 11 $35.00/0

DICOM - Hugo

DICOM Used for

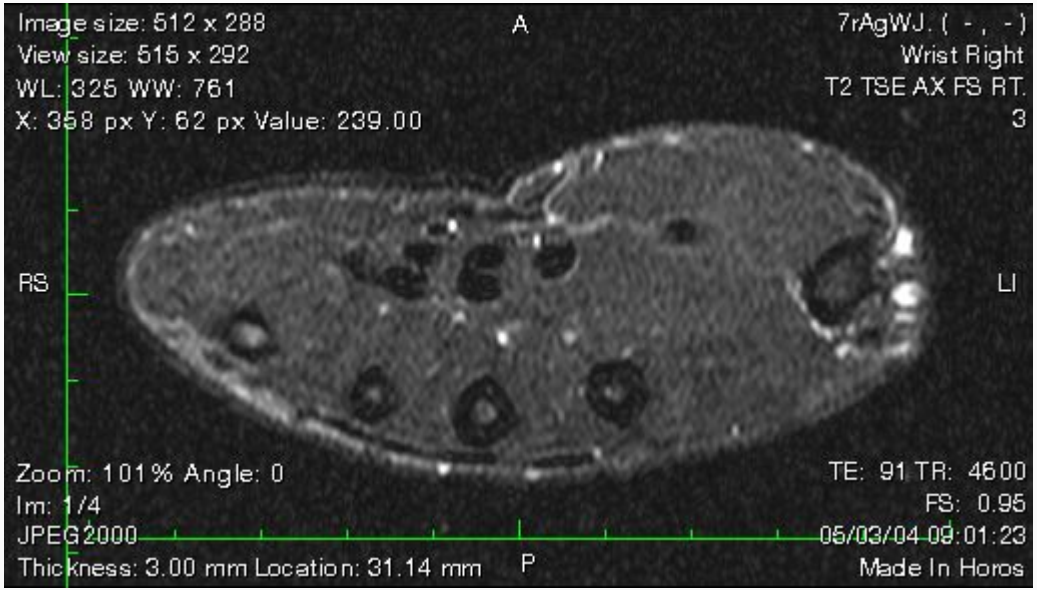
- Handling.
- Storing
- Printing
- Transmitting

→ Medical Images.

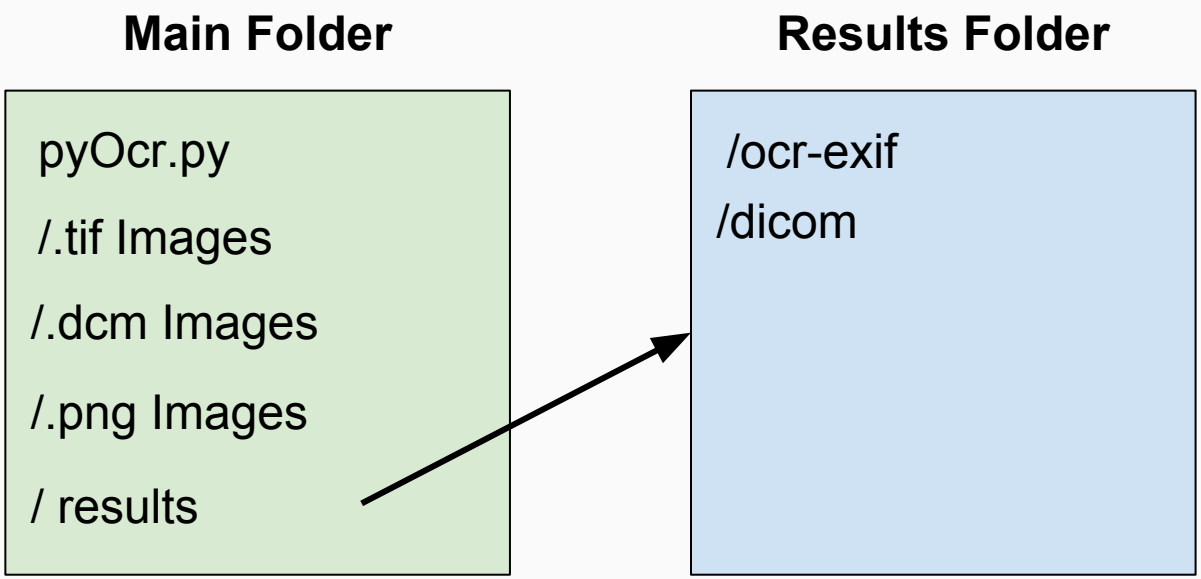
Developed using *Python* with the library *exifread*.

Main method used: *read_file*.

Sample Input for DICOM detection

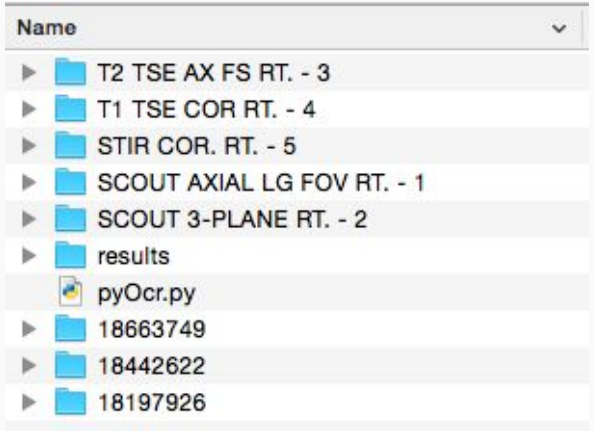


Program Structure

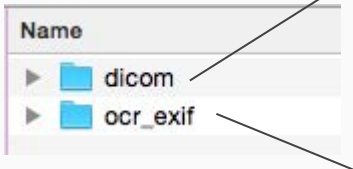


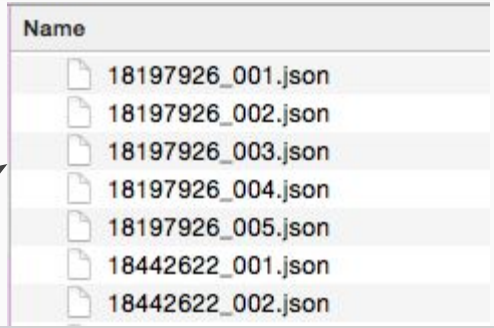
Program Structure

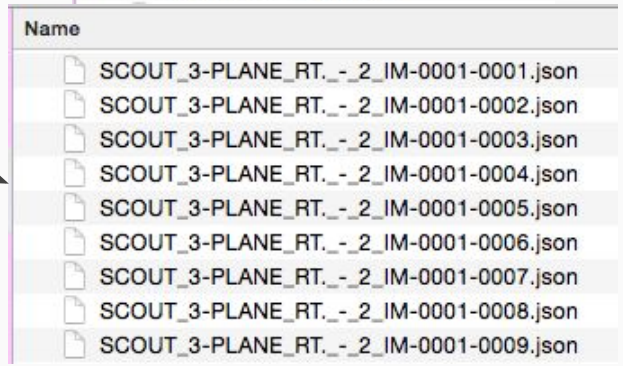
Main Folder



Results Folder







Output Json Format

```
1
2  {
3      "id": "folderName_FileName",
4
5      "text": "extractedOcrText",
6
7      "dcm": "extractedDcmText",
8
9      "low_level_features":"extractedLowLevelFeatures"
10 }
```

Low Level Features

Features used

- Brightness → *dark*, *balanced* or *bright*.
- Size → *small*, *medium* or *big*.
- Dominant color → *red*, *green* or *blue*.
- RGB composition → % of *red*, *green* and *blue*.

Developed using *Python* with the libraries skimage and PIL.

Stored in Json format.

Sample Input for low level features extraction


```
[
    {
        "dcm": "",
        "exif": "",
        "id": "pngbeagleA.png",
        "low_level_features": {
            "brightness": "balanced",
            "color_percentage": [
                36.452829257542504,
                39.63399505031924,
                23.913175692138253
            ],
            "dominant_color": "green",
            "size": "medium"
        },
        "text": ""
    }
]
```

Code Part 1

```python

#PyOcr.py
#Information Retrieval
#Created by Hugo Santana and Ignacio Amaya

from PIL import Image
import sys

import pyocr
import pyocr.builders
import codecs #Needed for UTF Conversion
import json #json converter library
import glob #Used to navigate throught system paths
import os   #Used to navigate throught system paths
import dicom #dicom library, do not use import pydicom
import exifread #exis library
```

```python
def start():
    tools = pyocr.get_available_tools()
    if len(tools) == 0:
        print("No OCR tool found")
        sys.exit(1)
    # The tools are returned in the recommended order of usage
    tool = tools[0]
    print("Will use tool '%s'" % (tool.get_name()))

    #Selecting language
    langs = tool.get_available_languages()
    #Using English
    lang = langs[0]


    #Going around all the files on the /files directory
    #Program should be ran on the root directory, all files should be on /files folder
    scanOcr_Exif(tool,lang)
    scanDicom(tool)
```

Code Part 2

```
18  def scanOcr_Exif(tool, lang):
19      for dirname in os.listdir(os.getcwd()):
20          for filename in glob.glob(os.path.join(dirname, '*.tif')):
21              print("file = " + filename)
22              #Getting text from image using ocr
23              txtOcr = tool.image_to_string(
24                  Image.open(filename),
25                  lang=lang,
26                  builder=pyocr.builders.TextBuilder()
27              )
28
29              #Getting text from image using exif
30              f=open(filename,'rb')
31              txtExif = str(exifread.process_file(f))
32              print(txtExif)
33              #Reusing file names as id.
34              filename=filename.replace("/", "_")
35              filename=filename.replace(".tif", "")
36
37              #Creating json Format
38              jsonString = json.dumps([{'id':filename,'text':txtOcr,'dcm':'','exif':txtExif}],
39
40              #Creating json file
41              with codecs.open("results/ocr_exif/"+filename + '.json', 'w') as outfile:
42                  outfile.write(jsonString)
43      return
```

Code Part 3

```
def scanDicom(tool):
    for dirname in os.listdir(os.getcwd()):
        for filename in glob.glob(os.path.join(dirname, '*.dcm')):
            print("file = "+filename)
            #Getting text from image using dicom
            txt = str(dicom.read_file(filename))
            #Reusing file names as id.
            filename=filename.replace("/", "_")
            filename=filename.replace(" ", "_")
            filename=filename.replace(".dcm", "")
            print(txt)
            #Creating json Format
            jsonString = json.dumps([{'id':filename,'text':'','dcm':txt,'exif':''}], sort_keys=True)

            #Creating json file
            with codecs.open("results/dicom/"+filename + '.json', 'w') as outfile:
                outfile.write(jsonString)
            print("Created id =" + filename)
    return
```

Code Part 4

```
from PIL import Image
import numpy as numpy
from  skimage import io,color,novice
import pylab as P
import glob
import json
import codecs
import os

#img_name = "tests/SREF.PNG"
#img_name1 = "tests/beagleA.png"

def get_low_level_features(filename):
    image = Image.open(filename)
    im = image.convert('L')
    im_array = numpy.array(im)

    #Brightness
    h,_,_ = P.hist(im_array.flatten(),bins=3,hold = False)
    brightness_value = max(enumerate(h),key=lambda x: x[1])
    brightness_dict = {0:'dark',1:'balanced',2:'bright'}
    brightness = brightness_dict[brightness_value[0]]
```

```
#Color
red = []
green = []
blue = []
for pixel in pic:
    red.append(pixel.red)
    green.append(pixel.green)
    blue.append(pixel.blue)
total_RGB = [sum(red),sum(green),sum(blue)]
color_percentage = [100*i/sum(total_RGB) for i in total_RGB]
color_percentage_labels='red percentage - {},green percentage - {},blue
percentage -
{}'.format(str(round(color_percentage[0],2)),str(round(color_percentage[1],2)),
str(round(color_percentage[2],2)))
dominant_color = max(enumerate(color_percentage),key=lambda x: x[1])
dominant_colors_dict= {0:'red',1:'green',2:'blue'}
dominant_color_label = dominant_colors_dict.get(dominant_color[0])
return 'brightness - {} , size - {} , dominant_color - {} , color percentage -
( {} )'.format(brightness,size,dominant_color_label,color_percentage_labels)
```

Code Part 5

```
#size
pic = novice.open(filename)
area = pic.size[0]*pic.size[1]
if area<=2500:
    size = 'small'
elif area > 2500 and area <= 900000:
    size = 'medium'
else:
    size = 'big'
```

```
for dirname in os.listdir(os.getcwd()):
    for filename in glob.glob(os.path.join(dirname,
    '*.PNG')):
        print(filename)
        low_level_dict = get_low_level_features(filename)
        #Reusing file names as id.
        filename=filename.replace("\\", "")
        filename=filename.replace("/", "_")
        filename=filename.replace(" ", "_")
        filename=filename.replace(".PNG", "")
        #Creating json Format
        jsonString =
        json.dumps([{'id':filename,'text':'','dcm':'','exif':
        '','low_level_features':low_level_dict}],
        sort_keys=True)
        print(filename)
        #Creating json file
        with codecs.open("results/png/"+filename + '.json',
        'w') as outfile:
            outfile.write(jsonString)
        print("Created id =" + filename)
```

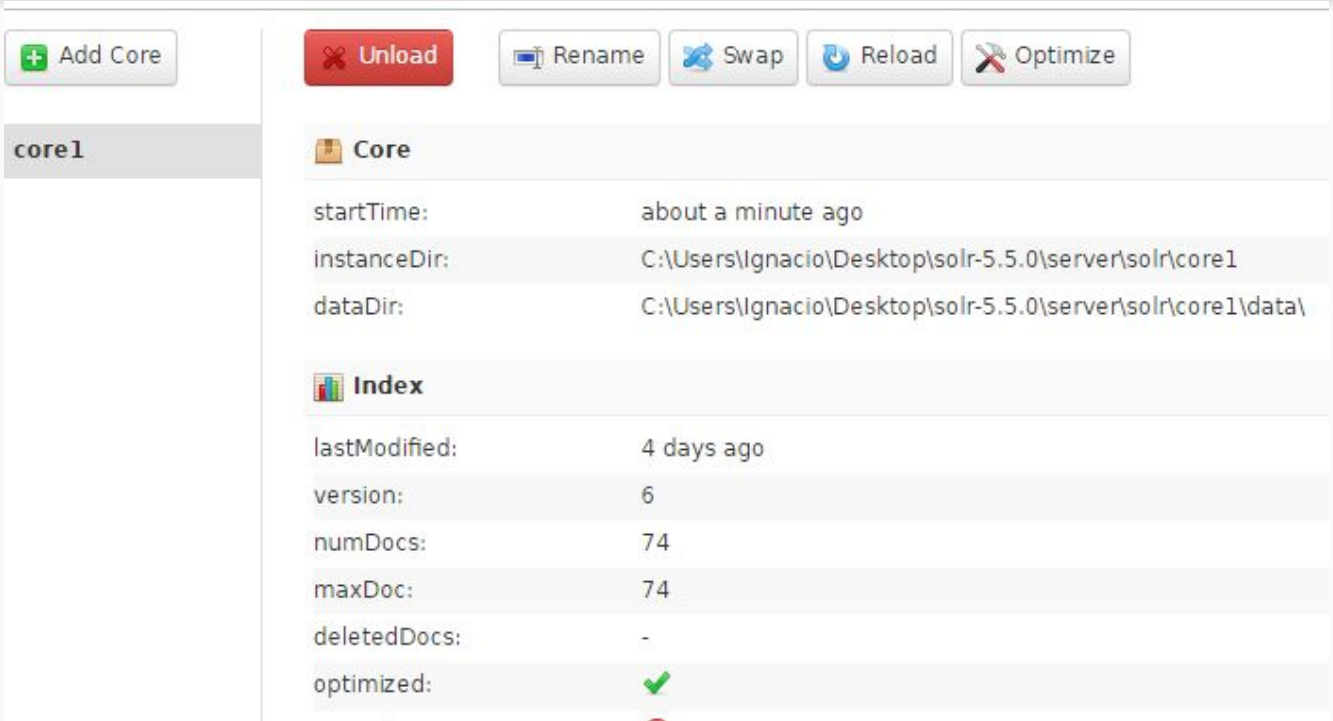
Apache Solr

- Open source enterprise search platform built on Apache Lucene.
- How it works
  - Installation
  - Starting solr → *.\solr.cmd start*
  - Creating core → *.\solr.cmd create -c coreName*
  - Accessing the UI → *http://localhost:8983/solr/*
- Python module requests used for automating uploading and querying.

Indexing into SolR

- Core name used: *core1*
- POST request used
- Number of documents uploaded (sample set of 84 images) :
  - TIFF: 15
  - DCM: 59
  - PNG: 10

Indexing into SolR



Indexing into SolR - Code

```
def upload_docs(files_folder,core_name):
    headers = {'content-type': 'application/json'}
    files_list = []
    for dirName, subdirList, fileList in os.walk(files_folder):
        for filename in fileList:
            if ".json" in filename.lower():  # check is the file is
            a json
                files_list.append(os.path.join(dirName,filename))
    for file in files_list:
        json_data=open(file).read()
        data = json.loads(json_data)
        r1 = requests.post('http://localhost:8983/solr/'+ core_name
        +'/update', data=json.dumps(data), headers=headers)
        if r1.ok != True:
            print('ERROR: the file {} could not be
            loaded'.format(file))
        else:
            print('File {} correctly uploaded to solr'.format(file))
```

Searching in Solr - Ignacio

Request-Handler (qt)

/select

— common —

q

The

fq

sort

start, rows

0 10

http://localhost:8983/solr/core1


```
{
  "responseHeader": {
    "status": 0,
    "QTime": 20,
    "params": {
      "indent": "true",
      "q": "The",
      "_": "1460732617444",
      "wt": "json"
    }
  },
  "response": {
    "numFound": 15,
    "start": 0,
    "docs": [
```

Searching in Solr

- One Python method implemented to automate the process of querying:

- Arguments
  - Search terms.
  - Terms retrieved.
  - Core to search in.

Searching in Solr - Code

```
def query_docs(input_text,output_fields,core_name):
    query = {'q': input_text, 'fl': output_fields, 'wt':'json',
    'indent':True}
    #gets XML responses
    r = requests.get('http://localhost:8983/solr/'+ core_name
    +'/select', params=query)
    if r.ok:
        json_text = json.loads(r.text)
        response = json_text['response']['docs']
    else:
        response = None
    return response
```

Use example

Folder: contains the jsons obtained before

Core_name: name of your core in Solr

```
upload_docs(folder,core_name)
#tests
r1 = query_docs('The',['id','text'],core_name)
r2 = query_docs(['ISO_IR 100'],['id','text','dcm'],core_name)
r3 = query_docs(['dark'],['id','low_level_features'],core_name)
```

The first parameter in *query_docs* is the term searched, while the second one is the fields you want to retrieve and the last one the core you want to look in.

Use example

```
r3 = query_docs(['dark'],['id','low_level_features'],core_name)
```


```
[
    {
        "id": "pngcuboA",
        "low_level_features": [
            "brightness - dark , size - medium , dominant_color - red ,
color percentage - ( red percentage - 33.72,green percentage - 32.99,blue percentage - 33.29 )"
        ]
    }
]
```

Any questions?