Report of Mid-Term Integration

Yuxiang Chen 5110309783

November 4, 2012

Contents

1 The Purpose of This Experiment and My Preparation

After reading the instructions in the demo ppt file, I find the main task for us to do is just set up a new search engine including both the function of text search and image search, with the framework modified by div+css. So I take some time on the Internet to learn how to use this method. Since the experiment is mainly based on lab4, I will try to make this report simple and short(Because I will also include the report of lab4 in the packaged homework).

2 The Main Part of the Experiment

The main part is nearly the same as lab4, including make indexes of both texts and images, then write the html files to show the result in the form of web pages.

2.1 Making an Index First

The index part is the same as lab3, so I'll only paste the codes here. And since we have to do two kinds of search, so I just make two indexes, one about text and the other about images. To make it easier to transport to the teach assistant, I will only crawl 30 of each web pages as a simple example. And you can use the

index code to crawl more web pages if you want. But since I crawled a web site about pictures of desktop last time, so this time my picture searching program will only concentrate on that site.
This is the index program about text index:

```
1  import sys, os, lucene, threading, time,chardet,urllib2
2  from datetime import datetime
3  from BeautifulSoup import BeautifulSoup
4  from ctypes import *
5  import re
6  import string
7
8  def filter_tags(htmlstr):
9      re_cdata=re.compile('//<!\[CDATA\[[^>]*//\]\]>',re.I)
10     re_script=re.compile('<\s*script[^>]*>[^<]*<\s*/\s*script\s*>',re.I)
11     re_style=re.compile('<\s*style[^>]*>[^<]*<\s*/\s*style\s*>',re.I)
12     re_br=re.compile('<br\s*?/?>')
13     re_h=re.compile('</?\w+[^>]*>')
14     re_comment=re.compile('<!--[^>]*-->')
15     s=re_cdata.sub('',htmlstr)
16     s=re_script.sub('',s)
17     s=re_style.sub('',s)
18     s=re_br.sub('\n',s)
19     s=re_h.sub('',s)
20     s=re_comment.sub('',s)
21     blank_line=re.compile('\n+')
22     s=blank_line.sub('\n',s)
23     s=replaceCharEntity(s)
24     return s
25
26
27 def replaceCharEntity(htmlstr):
28     CHAR_ENTITIES={'nbsp':' ','160':' ','lt':'<','60':'<','gt':'>','62':'>','amp
29     re_charEntity=re.compile(r'&#?(?P<name>\w+);')
30     sz=re_charEntity.search(htmlstr)
31     while sz:
32         entity=sz.group()
33         key=sz.group('name')
34         try:
35             htmlstr=re_charEntity.sub(CHAR_ENTITIES[key],htmlstr,1)
36             sz=re_charEntity.search(htmlstr)
37         except KeyError:
38             htmlstr=re_charEntity.sub('',htmlstr,1)
39             sz=re_charEntity.search(htmlstr)
40     return htmlstr
41
```

```
42
43 def repalce(s,re_exp,repl_string):
44     return re_exp.sub(repl_string,s)
45
46
47 class Ticker(object):
48
49     def __init__(self):
50         self.tick = True
51
52     def run(self):
53         while self.tick:
54             sys.stdout.write('.')
55             sys.stdout.flush()
56             time.sleep(1.0)
57
58 class IndexFiles(object):
59     """Usage: python IndexFiles <doc_directory>"""
60
61     def __init__(self, root, storeDir, analyzer):
62
63         if not os.path.exists(storeDir):
64             os.mkdir(storeDir)
65         store = lucene.SimpleFSDirectory(lucene.File(storeDir))
66         writer = lucene.IndexWriter(store, analyzer, True,
67                                     lucene.IndexWriter.MaxFieldLength.LIMITED)
68         writer.setMaxFieldLength(1048576)
69         self.indexDocs(root, writer)
70         ticker = Ticker()
71         print 'optimizing index',
72         threading.Thread(target=ticker.run).start()
73         writer.optimize()
74         writer.close()
75         ticker.tick = False
76         print 'done'
77
78     def indexDocs(self, root, writer):
79         for root, dirnames, filenames in os.walk(root):
80             for filename in filenames:
81                 if filename.endswith('.txt'):
82                     continue
83                 print "adding", filename
84                 try:
85                     path = os.path.join(root, filename)
86                     file = open(path)
87                     buf = file.read()
```

```
88                          contents=buf
89                          result = chardet.detect(buf)['encoding']
90                          if result=='GB2312':
91                              contents = buf.decode('gbk').encode('utf8')
92                          file.close()
93                          soup=BeautifulSoup(contents)
94                          url=mydict[filename]
95                          title=str(soup.head.title.string).decode('utf8')
96                          new_contents=filter_tags(contents)
97                          new_contents=str(new_contents).strip().decode('utf8')
98                          pos=new_contents.find('>')
99                          new_contents=new_contents[pos+1:]
100                         temp=new_contents.split()
101                         newtext=' '.join(temp)
102                         contents=''.join(soup.findAll(text=True))
103                         doc = lucene.Document()
104                         doc.add(lucene.Field("text", newtext,
105                                              lucene.Field.Store.YES,
106                                              lucene.Field.Index.NOT_ANALYZED))
107                         doc.add(lucene.Field("url", url,
108                                              lucene.Field.Store.YES,
109                                              lucene.Field.Index.NOT_ANALYZED))
110                         doc.add(lucene.Field("title", title,
111                                              lucene.Field.Store.YES,
112                                              lucene.Field.Index.NOT_ANALYZED))
113                         if len(contents) > 0:
114                             dll=cdll.LoadLibrary("F:\\ICTCLAS50_Windows_32_C\ICTCLAS
115                             dll.ICTCLAS_Init(c_char_p("F:\\ICTCLAS50_Windows_32_C"))
116                             strlen = len(c_char_p(contents).value)
117                             t =c_buffer(strlen*6)
118                             bSuccess = dll.ICTCLAS_ParagraphProcess(c_char_p(content
119                             contents=t.value.decode('gbk').encode('utf8')
120                             ##list=t.value.split()
121                             ##print ' '.join(list)
122                             dll.ICTCLAS_Exit()
123                             doc.add(lucene.Field("contents", contents,
124                                                  lucene.Field.Store.NO,
125                                                  lucene.Field.Index.ANALYZED))
126                         else:
127                             print "warning: no content in %s" % filename
128                         writer.addDocument(doc)
129                 except Exception, e:
130                     print "Failed in indexDocs:", e
131
132 if __name__ == '__main__':
133 ##     if len(sys.argv) < 2:
```

```
##          print IndexFiles.__doc__
##          sys.exit(1)
    lucene.initVM()
    print 'lucene', lucene.VERSION
    start = datetime.now()
    dic= open('F:\\html\index.txt')
    d = dic.readlines()
    dic.close()
    mydict = {}
    for word in d:
        value=''
        try:
            key = word.split(';')[0]
            value = word.split(';')[1]
            mydict[key] = value
        except:
            pass
    try:
##          IndexFiles(sys.argv[1], "index", lucene.SimpleAnalyzer(lucene.Version.
        IndexFiles('F:\\html', "F:\\index", lucene.SimpleAnalyzer(lucene.Version
        end = datetime.now()
        print end - start
    except Exception, e:
        print "Failed: ", e
```

But this time the index about texts is a little different from lab3, but is the same as lab4, of course. Since I do write some codes to avoid the appearance of some useless html tags in the text I stored. You can see it clearly in my report of lab4. And this is the index program about image index:

```
import sys, os, lucene, threading, time,chardet,urllib2,re
from datetime import datetime
from BeautifulSoup import BeautifulSoup
from ctypes import *
import urllib
import Queue
import urlparse

#                                       http://www.ommoo.com/

"""
This class is loosely based on the Lucene (java implementation) demo class
org.apache.lucene.demo.IndexFiles.  It will take a directory as an argument
and will index all of the files in that directory and downward recursively.
It will index on the file path, the file name and the file contents.
The
```

```
16 resulting Lucene index will be placed in the current directory and called
17 'index'.
18 """
19
20 class Ticker(object):
21
22     def __init__(self):
23         self.tick = True
24
25     def run(self):
26         while self.tick:
27             sys.stdout.write('.')
28             sys.stdout.flush()
29             time.sleep(1.0)
30
31 class IndexFiles(object):
32     """Usage: python IndexFiles <doc_directory>"""
33
34     def __init__(self, root, storeDir, analyzer):
35
36         if not os.path.exists(storeDir):
37             os.mkdir(storeDir)
38         store = lucene.SimpleFSDirectory(lucene.File(storeDir))
39         writer = lucene.IndexWriter(store, analyzer, True,
40                                     lucene.IndexWriter.MaxFieldLength.LIMITED)
41         writer.setMaxFieldLength(1048576)
42         self.indexDocs(root, writer)
43         ticker = Ticker()
44         print 'optimizing index',
45         threading.Thread(target=ticker.run).start()
46         writer.optimize()
47         writer.close()
48         ticker.tick = False
49         print 'done'
50
51     def indexDocs(self, root, writer):
52         for root, dirnames, filenames in os.walk(root):
53             for filename in filenames:
54                 if filename.endswith('.txt'):
55                     continue
56                 print "adding", filename
57                 try:
58                     path = os.path.join(root, filename)
59                     file = open(path)
60                     buf = file.read()
61                     contents=buf
```

```
62                      result = chardet.detect(buf)['encoding']
63                      if result=='GB2312':
64                          contents = buf.decode('gbk').encode('utf8')
65                      file.close()
66                      soup=BeautifulSoup(contents)
67                      url=mydict[filename]
68                      proto, rest = urllib.splittype(url)
69                      site, rest = urllib.splithost(rest)
70                      title=str(soup.head.title.string.strip()).decode('utf8')
71                      flag2=0
72                      for i in soup.findAll('img'):
73                          contents=""
74                          flag1=0
75                          flag3=0
76                          try:
77                              contents=contents+' '+i['alt']
78                          except:
79                              pass
80                          tempurl=i['src']
81                          imgurl=urlparse.urljoin(url,tempurl)
82                          temp=i.parent.parent
83                          try:
84                              photoid=temp.find('a')['data-photo-id']
85                              flag1=1
86                          except:
87                              pass
88                          try:
89                              picid=temp.parent.find('article')['id']
90                              flag3=1
91                          except:
92                              pass
93                          try:
94                              for t in temp.findAll('b'):
95                                  try:
96                                      contents=contents+' '+t.string.strip()
97                                  except:
98                                      pass
99                          except:
100                             pass
101                         try:
102                             for k in temp.findAll('p'):
103                                 try:
104                                     contents=contents+' '+k.string.strip()
105                                 except:
106                                     pass
107                         except:
```

```
                    pass
                try:
                    for j in temp.findAll('span',{'class':'title'}):
                        try:
                            contents=contents+' '+j.string.strip()
                        except:
                            pass
                except:
                    pass
                if flag1==1:
                    timetowait=0
                    try:
                        for p in temp.parent.findAll('div',{'class':'car
                            if timetowait<flag2:
                                timetowait+=1
                                continue
                            contents=contents+' '+p.string.strip()
                            flag2+=1
                            break
                    except:
                        pass
                if flag3==1:
                    try:
                        for q in temp.parent.findAll('div',{'class':'pos
                            r=q.find('h1')
                            contents=contents+' '+str(r.string).decode('
                            break
                    except:
                        pass
                contents=contents.strip()
                doc = lucene.Document()
                doc.add(lucene.Field("imgurl", imgurl,
                                     lucene.Field.Store.YES,
                                     lucene.Field.Index.NOT_ANALYZED))
                doc.add(lucene.Field("url", url,
                                     lucene.Field.Store.YES,
                                     lucene.Field.Index.NOT_ANALYZED))
                doc.add(lucene.Field("title", title,
                                     lucene.Field.Store.YES,
                                     lucene.Field.Index.NOT_ANALYZED))
                if len(contents) > 0:
                    dll=cdll.LoadLibrary("F:\\ICTCLAS50_Windows_32_C\ICT
                    dll.ICTCLAS_Init(c_char_p("F:\\ICTCLAS50_Windows_32_
                    strlen = len(c_char_p(contents).value)
                    t =c_buffer(strlen*6)
                    bSuccess = dll.ICTCLAS_ParagraphProcess(c_char_p(con
```

```
                                    contents=t.value.decode('gbk').encode('utf8')
                                    ##list=t.value.split()
                                    ##print ' '.join(list)
                                    dll.ICTCLAS_Exit()
                                    doc.add(lucene.Field("contents", contents,
                                                         lucene.Field.Store.NO,
                                                         lucene.Field.Index.ANALYZED))
                                else:
                                    print "warning: no content in part of %s" % filename
                                writer.addDocument(doc)
                        except Exception, e:
                            print "Failed in indexDocs:", e

if __name__ == '__main__':
##    if len(sys.argv) < 2:
##        print IndexFiles.__doc__
##        sys.exit(1)
    lucene.initVM()
    print 'lucene', lucene.VERSION
    start = datetime.now()
    dic= open('F:\\html\index.txt')
    d = dic.readlines()
    dic.close()
    mydict = {}
    for word in d:
        key = word.split(';')[0]
        value = word.split(';')[1]
        mydict[key] = value
    try:
##        IndexFiles(sys.argv[1], "index", lucene.WhitespaceAnalyzer(lucene.Vers
        IndexFiles('F:\\html', "F:\\imgindex", lucene.WhitespaceAnalyzer(lucene.
        end = datetime.now()
        print end - start
    except Exception, e:
        print "Failed: ", e
```

And the pictures of the indexes have been given in report of lab4, so I won't present it here.

2.2 The Part of Establishing the Web Page

In this part, what I need to do is to modify my html files in the form of div+css. First, we need to change the image search and text search at first. And since the formtest html files are really easy, so I use the normal html files in them. And I use the div+css in the result html files. It is quite easy if we know how to use css selector to edit the properties of the contents in it and how to insert

python clause in it.
Now here is the file of text search 'formtest.html':

```
$def with(form)
        <title>Chris        </title>

        <a href="/i" >    </a>
                <form name="input" form action="/s" method="GET">          :
                <input type="keyword" name="keyword" />
                <input type="submit" value="          "/>
                </form>
```

And this is the file of image search 'img formtest':

```
$def with(form)
    <title>Chris        </title>
    <a href="/" >    </a>

        <form name="input" form action="/im" method="GET">          :
        <input type="keyword" name="keyword" />
        <input type="submit" value="          "/>
        </form>
```

This is the file of text search 'search.html', and you can see how I edit the pages in this file.

```
$def with (form,name,qurl,qtitle,qnew_text1,qnew_text2,q_query,total)
<!DOCTYPE html>
<html>
        <head>
        <title>Chris        </title>

        <a href="/i" >    </a>
        <form name="input" form action="/s" method="GET">          :
                <input type="keyword" name="keyword" />
                <input type="submit" value="          "/>
        </form>
        <style type="text/css">
                #red {color:red;}
                #green {color:green;}
                [url] {color:green;}
                div.container{width:100%;margin:5px;border:3px solid purple;line
                div.headcontainer{width:100%;margin:5px;border:3px solid purple;
                div.content{margin:0px;padding:1em;}
        </style>
        </head>
        <body>
                <div class="headcontainer"><h2>                "␣$name␣"              $
```

```
                $if total:
                        $for i in range(total):
                        <div class="container">
                                <div class="content">
                                        <h2><a href="$qurl[i]">$qtitle[i]</a><
                                        <p>$qnew_text1[i].strip()<span id='red'>
                                        <p url="$qurl[i]">$qurl[i]</p>
                                </div>
                        </div>
                $else:
                        <div class="container">
                                <div class="content">
                                        <h2>
                                </div>
                        </div>
        </body>
</html>
```

Now is the last part, I write the file of image search 'img result.html' in this way. And I will explain how I design this page in next section.

```
$def with (form,name,qurl,qtitle,qimgurl,total)
<!DOCTYPE html>
<html>
        <head>
        <title>Chris        </title>
        <a href="/" >    </a>

        <form name="input" form action="/im" method="GET">          :
                <input type="keyword" name="keyword" />
                <input type="submit" value="          "/>
        </form>
        <style type="text/css">
                div.container{width:100%;margin:5px;border:3px solid blue;line-h
                div.scontainer{width:320px;height:240px;margin:5px;border:3px so
                div.content{margin:0px;padding:3px;max-height:220px}
        </style>
        </head>
        <body>
                <div class="container"><div class="content"><h2>                "$
                $if total:
                        $for i in range(total):
                                <div class="scontainer">
                                        <div class="content">
                                                <h4><a href="$qurl[i]">$qtitle
                                                <img src="$qimgurl[i]"></div><
                $else:
```

```
27                          <div class="container">
28                                  <div class="content">
29                                  <h2>
30                                  </div>
31                          </div>
32                  </div>
33          </body>
34  </html>
```

And I will give the screenshots of my search engine together in the search section.

3 The Main Searching Part of the Program

This part is simply an addition to lab4, only attaching 2 classes and a function about image search, which is nearly the same as text search. And with these functions, we are able to search for certain pictures and show them out in a web page.

```
1  import web
2  from web import form
3  import urllib2
4  import os
5  from lucene import \
6      QueryParser, IndexSearcher, SimpleAnalyzer, SimpleFSDirectory, File, \
7      VERSION, initVM, Version
8  from ctypes import *
9
10 urls = (
11     '/', 'index',
12     '/s', 'text',
13     '/i', 'img_index',
14     '/im', 'img'
15 )
16
17
18 render = web.template.render('templates') # your templates
19
20 login = form.Form(
21     form.Textbox('keyword'),
22     form.Button('Search'),
23 )
24
25 def func(command, searcher, analyzer):
26     if len(command) > 0:
27         dll=cdll.LoadLibrary("F:\\ICTCLAS50_Windows_32_C\ICTCLAS50.dll")
28         dll.ICTCLAS_Init(c_char_p("F:\\ICTCLAS50_Windows_32_C"))
```

```
29          strlen = len(c_char_p(command).value)
30          t =c_buffer(strlen*6)
31          bSuccess = dll.ICTCLAS_ParagraphProcess(c_char_p(command),c_int(strlen),
32          command=t.value.decode('gbk').encode('utf8')
33          ##list=t.value.split()
34          ##print ' '.join(list)
35          dll.ICTCLAS_Exit()
36          command=command.decode('utf8')
37      query = QueryParser(Version.LUCENE_CURRENT, "contents",analyzer).parse(comma
38      scoreDocs = searcher.search(query, 50).scoreDocs
39      total=len(scoreDocs)
40      qtitle=[]
41      qurl=[]
42      qnew_text1=[]
43      qnew_text2=[]
44      q_query=[]
45      new_query=str(query).replace ('contents:','').decode('utf8')
46      if total==0:
47          return command,qurl,qtitle,qnew_text1,qnew_text2,q_query,total
48      for scoreDoc in scoreDocs:
49          doc = searcher.doc(scoreDoc.doc)
50          text=doc.get("text")
51          new_text=str(text).decode('utf8')
52          temp_query=new_query.replace('␣','')
53          num=new_text.find(temp_query)
54          query_len=len(temp_query)
55          splited_query=new_query.split('␣')
56          splitlen=len(splited_query)
57          if (num!=-1):
58              try:
59                  new_text1=str(new_text[num-30:num]).strip().decode('utf8','ignor
60                  new_text2=str(new_text[num+query_len:num+30+query_len]).strip().
61              except:
62                  try:
63                      new_text1=str(new_text[num-30:num]).strip().decode('utf8','i
64                      new_text2=""
65                  except:
66                      try:
67                          new_text1=""
68                          new_text2=str(new_text[num+query_len:num+30+query_len]).
69                      except:
70                          new_text1=""
71                          new_text2=""
72              q_query.append(temp_query)
73          else:
74              for i in range(splitlen):
```

```
75                  parted_query=splited_query[i].decode('utf8')
76                  num=new_text.find(parted_query)
77                  if num==-1:
78                      continue
79                  query_len=len(parted_query)
80                  try:
81                      new_text1=str(new_text[num-30:num]).strip().decode('utf8','ig
82                      new_text2=str(new_text[num+query_len:num+30+query_len]).strip
83                  except:
84                      try:
85                          new_text1=str(new_text[num-30:num]).strip().decode('utf8
86                          new_text2=""
87                      except:
88                          try:
89                              new_text1=""
90                              new_text2=str(new_text[num+query_len:num+30+query_le
91                          except:
92                              new_text1=""
93                              new_text2=""
94                  q_query.append(parted_query)
95                  break
96              if num==-1:
97                  total=total-1
98                  continue
99          title=doc.get("title")
100         url=doc.get("url")
101         qurl.append(url)
102         qtitle.append(title)
103         qnew_text1.append(new_text1)
104         qnew_text2.append(new_text2)
105     return command,qurl,qtitle,qnew_text1,qnew_text2,q_query,total
106
107 def img_func(command, searcher, analyzer):
108     if len(command) > 0:
109         dll=cdll.LoadLibrary("F:\\ICTCLAS50_Windows_32_C\ICTCLAS50.dll")
110         dll.ICTCLAS_Init(c_char_p("F:\\ICTCLAS50_Windows_32_C"))
111         strlen = len(c_char_p(command).value)
112         t =c_buffer(strlen*6)
113         bSuccess = dll.ICTCLAS_ParagraphProcess(c_char_p(command),c_int(strlen),
114         command=t.value.decode('gbk').encode('utf8')
115         ##list=t.value.split()
116         ##print ' '.join(list)
117         dll.ICTCLAS_Exit()
118         command=command.decode('utf8')
119     if command == '':
120         return
```

```
121     query = QueryParser(Version.LUCENE_CURRENT, "contents",analyzer).parse(comma
122     scoreDocs = searcher.search(query, 50).scoreDocs
123     total=len(scoreDocs)
124     qtitle=[]
125     qurl=[]
126     qimgurl=[]
127     for scoreDoc in scoreDocs:
128         doc = searcher.doc(scoreDoc.doc)
129         imgurl=doc.get("imgurl")
130         if imgurl not in qimgurl:
131             title=doc.get("title")
132             qtitle.append(title)
133             url=doc.get("url")
134             qurl.append(url)
135             qimgurl.append(imgurl)
136         else:
137             total-=1
138             continue
139     return command,qurl,qtitle,qimgurl,total
140
141 class index:
142     def GET(self):
143         f = login()
144         return render.formtest(f)
145
146 class img_index:
147     def GET(self):
148         f = login()
149         return render.img_formtest(f)
150
151 class text:
152     def GET(self):
153         vm_env = initVM()
154         form1 = login()
155         user_data = web.input()
156         vm_env.attachCurrentThread()
157         STORE_DIR = "F:\\index"
158         directory = SimpleFSDirectory(File(STORE_DIR))
159         searcher = IndexSearcher(directory, True)
160         analyzer = SimpleAnalyzer(Version.LUCENE_CURRENT)
161         a,b,c,d,e,f,g= func(user_data.keyword,searcher,analyzer)
162         searcher.close()
163         return render.result(form1,a,b,c,d,e,f,g)
164
165
166 class img:
```

```
167     def GET(self):
168         vm_env = initVM()
169         form1 = login()
170         user_data = web.input()
171         vm_env.attachCurrentThread()
172         STORE_DIR = "F:\\imgindex"
173         directory = SimpleFSDirectory(File(STORE_DIR))
174         searcher = IndexSearcher(directory, True)
175         analyzer = SimpleAnalyzer(Version.LUCENE_CURRENT)
176         a,b,c,d,e = img_func(user_data.keyword,searcher,analyzer)
177         searcher.close()
178         return render.img_result(form1,a,b,c,d,e)
179
180 if __name__ == "__main__":
181     app = web.application(urls, globals())
182     app.run()
```

And now I will show you some pictures of my finished search engine.

Figure 1: text search 1

4 The Problems I Met and My Thoughts

In this part, the main problem is how to build my pages using div+css. At first, I find I couldn't change the searching pages about text and images, then with the help of TA, I find it was because of this part:

```
1 urls = (
2     '/', 'index',
3     '/s', 'text',
4     '/i', 'img_index',
5     '/im', 'img')
```

Figure 2: text search 2

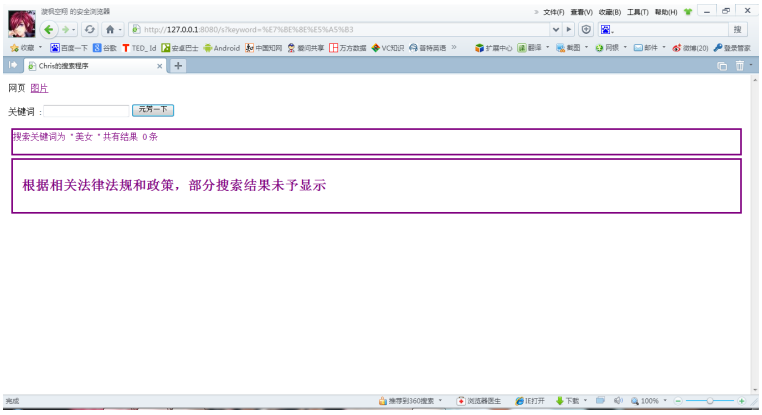

Figure 3: text search 3

Figure 4: image search 1

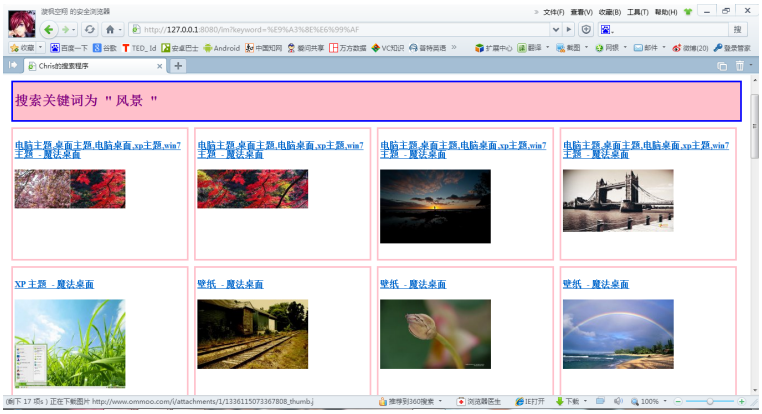

Figure 5: image search 2

Figure 6: image search 3

When I write it at the first time, I left a comma in the third line, leading to a series of errors. And after this, it seems easier. I finished the text searching part easily. But at the image searching part, I met a trouble at first: I couldn't print some pictures in a same line. Then after some research, I find it is because this property :"float:left", which allows elements to be shown on its left.
And in the index part, I also have some trouble getting rid of the html tags and other useless codes, and I search for it on the Internet, finding it can be solved using regular expression:

```
1  def filter_tags(htmlstr):
2      re_cdata=re.compile('//<!\[CDATA\[[^>]*//\]\]>',re.I)
3      re_script=re.compile('<\s*script[^>]*>[^<]*<\s*/\s*script\s*>',re.I)
4      re_style=re.compile('<\s*style[^>]*>[^<]*<\s*/\s*style\s*>',re.I)
5      re_br=re.compile('<br\s*?/?>')
6      re_h=re.compile('</?\w+[^>]*>')
7      re_comment=re.compile('<!--[^>]*-->')
8      s=re_cdata.sub('',htmlstr)
9      s=re_script.sub('',s)
10     s=re_style.sub('',s)
11     s=re_br.sub('\n',s)
12     s=re_h.sub('',s)
13     s=re_comment.sub('',s)
14     blank_line=re.compile('\n+')
15     s=blank_line.sub('\n',s)
16     s=replaceCharEntity(s)
17     return s
18
19
20 def replaceCharEntity(htmlstr):
21     CHAR_ENTITIES={'nbsp':' ','160':' ','lt':'<','60':'<','gt':'>','62':'>','amp
22     re_charEntity=re.compile(r'&#?(?P<name>\w+);')
23     sz=re_charEntity.search(htmlstr)
24     while sz:
25         entity=sz.group()
26         key=sz.group('name')
27         try:
28             htmlstr=re_charEntity.sub(CHAR_ENTITIES[key],htmlstr,1)
29             sz=re_charEntity.search(htmlstr)
30         except KeyError:
31             htmlstr=re_charEntity.sub('',htmlstr,1)
32             sz=re_charEntity.search(htmlstr)
33     return htmlstr
```

So after solving all these problems and set the max width of the pictures, I successfully finished my html file.
But I also know there is still a lot of things to revise. For example, I can divide the outcomes into several pages, or add some more search like music search, etc.

But I'm really busy this time, so maybe I will do more about this in the future.