ENGR 102 PROGRAMMING PRACTICE

WEEK 11

İSTANBUL ŞEHİR UNIVERSITY

Searching & Ranking

Search Engine

1. **Crawl to collect documents.**
2. Index to improve search.
3. Query for a select set of documents.
4. Rank the documents

Crawling pages

```
def crawl(self,pages,depth=2):
    for i in range(depth):
      newpages={}
      for page in pages:
        c = urllib2.urlopen(page)
        soup = BeautifulSoup(c.read())
        if not self.addtoindex(page,soup):
             continue

        links = soup('a')

        outgoingLinkCount = 0
        for link in links:
          if ('href' in dict(link.attrs)):
            url=urljoin(page,link['href'])

            if not self.isindexed(url):
              newpages.append(url)

            linkText = self.gettextonly(link)
            added = self.addlinkref(page,url,linkText)
            if added : outgoingLinkCount += 1
        self.urllist[smart_str(page)] = outgoingLinkCount
      pages=newpages
```

Search Engine

1. Crawl to collect documents.
2. **Index to improve search.**
3. Query for a select set of documents.

Setting Up Database

Four dictionaries:

- **urllist** is the list of URLs that have been indexed.
  {url: outgoing_link_count}
- **wordlocation** is a list of the locations of words in the documents.
  {word: {url: [loc1, loc2, ..., locN]}}
- **link** stores two URL IDs, indicating a link from one page to another.
  {tourl: {fromUrl: None}}
- **linkwords** store words that are included in a link.
  {word: [(urlFrom1, urlTo1), ..., (urlFromN, urlToN)]}

Building the Database

- The database will be stored using `shelve` module
- Provides persistent object storage on disk
- Similar to anydbm, but more practical
- Use with `import shelve`

shelve – Persistent storage of arbitrary Python objects

- Key-value structure (like a dictionary)
- Persists data on disk (like anydbm)
- Keys may only be strings (like anydbm)
- Values may be any object (unlike anydbm, like a dictionary)
    - No need to pickle objects
- Handles updates automagically

shelve – open and insert data

```
import shelve

s = shelve.open('test_shelf.db')
s['key1'] = {'int': 10, 'float':9.5, 'string':'data'}
s.close()

# this will create test_shelf.db file on disk
```

shelve – read existing content

```
import shelve


s = shelve.open('test_shelf.db')
existing = s['key1']
print existing
s.close()
# prints: {'int': 10, 'float': 9.5, 'string': 'data'}
```

shelve – auto update with writeback = True

```
import shelve

s = shelve.open('test_shelf.db')
print s['key1']
s['key1']['new_value'] = 'this was not here before'
s.close()

s = shelve.open('test_shelf.db')
print s['key1']
s.close()
# prints: {'int': 10, 'float': 9.5, 'string': 'data'}
          {'int': 10, 'float': 9.5, 'string': 'data'}
```

shelve – auto update with writeback = True

```
import shelve

s = shelve.open('test_shelf.db', writeback = True)
print s['key1']
s['key1']['new_value'] = 'this was not here before'
s.close()
# prints: {'int': 10, 'float': 9.5, 'string': 'data'}

s = shelve.open('test_shelf.db', writeback = True)
print s['key1']
s.close()
# prints: {'int': 10, 'new_value': 'this was not here
#            before', 'float': 9.5, 'string': 'data'}
```

Setting Up the Database

```
import mysearchengine
pagelist = ['http://sehir.edu.tr']
dbtables = {'urllist': 'urllist.db',
            'wordlocation': 'wordlocation.db',
            'link': 'link.db',
            'linkwords': 'linkwords.db'}
crawler = mysearchengine.crawler(dbtables)
crawler.createindextables()
```

createindextables( )

```
# Create the database tables
def createindextables(self):
    # {url:outgoing_link_count}
    self.urllist = shelve.open(self.dbtables['urllist'], writeback=True, flag='c')

    #{word:{url:[loc1, loc2, ..., locN]}}
    self.wordlocation = shelve.open(self.dbtables['wordlocation'], writeback=True, flag='c')

    #{tourl:{fromUrl:None}}
    self.link = shelve.open(self.dbtables['link'], writeback=True, flag='c')

    #{word:[(urlFrom, urlTo), (urlFrom, urlTo), ..., (urlFrom, urlTo)]}
    self.linkwords = shelve.open(self.dbtables['linkwords'], writeback=True, flag='c')
```

Adding to the Database

1. Get a list of words on the page.
2. Add the page and all the words to the index.
3. Create links between them with their locations in the document.

Finding Words on a Page

- The files on the Web are HTML and thus contain a lot of tags, properties, etc.
- The first step is to extract all the parts of the page that are text.
- You can do this by searching the soup for text nodes and collecting all their content.

Problem

- Write a function **`gettextonly`** that
  - inputs a tag and
  - outputs a string composed of all text under the tag
    - excluding html tags, properties, ...
- See the file gettextonly.py

```
html_doc = """
<html>
    <head><title>The Dormouse's story</title></head>
    <body><p><b>    Story starts... and ends</b></p></body>
</html>
"""
root_tag = BeautifulSoup(html_doc, "html.parser")
gettextonly(root_tag)
```

Output:

```
The Dormouse's story

Story starts... and ends
```

Finding the Words

- Split a string into a list of separate words so that they can be added to the index.
- Our approach:
  - Consider anything that isn't a letter or a number to be a separator.
- You can do this using a regular expression.

Separating into Words

```
# Separate the words by any non-alphanumeric character
def separatewords(self, text):                 [^a-zA-Z0-9_]
  splitter = re.compile(r'\W+')
  return [s.lower() for s in splitter.split(text)
                                            if s != '']
```

Crawling pages

```
def crawl(self,pages,depth=2):
    for i in range(depth):
      newpages={}
      for page in pages:
        c = urllib2.urlopen(page)
        soup = BeautifulSoup(c.read())
        if not self.addtoindex(page,soup):
             continue

        links = soup('a')

        outgoingLinkCount = 0
        for link in links:
          if ('href' in dict(link.attrs)):
            url=urljoin(page,link['href'])

            if not self.isindexed(url):
              newpages.append(url)

            linkText = self.gettextonly(link)
            added = self.addlinkref(page,url,linkText)
            if added : outgoingLinkCount += 1
        self.urllist[smart_str(page)] = outgoingLinkCount
      pages=newpages
```

Checking if this page is already indexed

```
# Return true if this url is already indexed
def isindexed(self, url):
    cleaned_url = smart_str(url)
    # urllist = {url:outgoing_link_count}
    if cleaned_url in self.urllist:
        return True
    else:
        return False
```

Adding into the index

```
# Index an individual page
def addtoindex(self, url, soup):
    if self.isindexed(url):
        print 'skip', url, 'already indexed'
        return False

    print 'Indexing ' + url

    # Get the individual words
    text = self.gettextonly(soup)
    words = self.separatewords(text)

    # Record each word found on this page
    for i in range(len(words)):
        word = smart_str(words[i])

        if word in ignorewords:
            continue

        self.wordlocation.setdefault(word, {})

        self.wordlocation[word].setdefault(url, [])
        self.wordlocation[word][url].append(i)
    return True
```

Recording links

```
# Add a link between two pages
def addlinkref(self, urlFrom, urlTo, linkText):
    fromUrl = smart_str(urlFrom)
    toUrl = smart_str(urlTo)

    if fromUrl == toUrl: return False

    self.link.setdefault(toUrl, {})
    # In case fromUrl already recorded, return false.
    if fromUrl in self.link[toUrl]:
        return False

    self.link[toUrl][fromUrl] = None

    words = self.separatewords(linkText)
    for word in words:
        word = smart_str(word)

        if word in ignorewords: continue

        self.linkwords.setdefault(word, [])
        self.linkwords[word].append((fromUrl, toUrl))

    return True
```

Crawling pages

```
import mysearchengine
pagelist = ['http://sehir.edu.tr']
dbtables = {'urllist': 'urllist.db',
            'wordlocation': 'wordlocation.db',
            'link': 'link.db',
            'linkwords': 'linkwords.db'}
crawler = mysearchengine.crawler(dbtables)
crawler.createindextables()
crawler.crawl(pagelist)
```