İSTANBUL ŞEHİR UNIVERSITY

ENGR 212: Programming Practice

Week 10

İSTANBUL ŞEHİR UNIVERSITY

Searching and Ranking

Adding to the Database

İSTANBUL ŞEHİR UNIVERSITY

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

Finding Words on a Page

İSTANBUL ŞEHİR UNIVERSITY

- The files on the Web are HTML and thus contain a lot of tags, properties, and other information that doesn't belong in the database.
- 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.

Finding the text on a page

İSTANBUL ŞEHİR UNIVERSITY

```
# Extract the text from an HTML page (no tags)
def gettextonly(self, soup):
    v = soup.string
    if v == None:
        c = soup.contents
        resulttext = ''
        for t in c:
            subtext = self.gettextonly(t)
            resulttext += subtext + '\n'
        return resulttext
     else:
          return v.strip()
```

Finding the Words on a Page

İSTANBUL ŞEHİR UNIVERSITY

- Next is to split a string into a list of separate words so that they can be added to the index.
- There has been a lot of research on this.
- 1st attempt: 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

İSTANBUL ŞEHİR ÜNİVERSİTY

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

Crawling pages

İSTANBUL ŞEHİR UNIVERSITY

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

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

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

            linkText=self.gettextonly(link)
            self.addlinkref(page,url,linkText)


      pages=newpages
```

Checking if this page is already indexed

İSTANBUL ŞEHİR UNIVERSITY

```
# Return true if this url is already indexed
def isindexed(self, url):
    # urllist = {url:outgoing_link_count}
    if not self.urllist.has_key(smart_str(url)):
        return False
    else:
        return True
```

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

İSTANBUL ŞEHİR UNIVERSITY

```
# 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, {})

    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

İSTANBUL ŞEHİR UNIVERSITY

```
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)
```

Search Engine

İSTANBUL ŞEHİR UNIVERSITY

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

Querying

İSTANBUL ŞEHİR UNIVERSITY

```
def getmatchingpages(self,q):
    results = {}
    # Split the words by spaces
    words = [smart_str(word) for word in q.split()]
    if words[0] not in self.wordlocation:
        return results, words

    url_set = set(self.wordlocation[words[0]].keys())

    for word in words[1:]:
        if word not in self.wordlocation:
            return results, words
        url_set = url_set.intersection(self.wordlocation[word].keys())

    for url in url_set:
        results[url] = []
        for word in words:
            results[url].append(self.wordlocation[word][url])

    return results, words
```

Querying

İSTANBUL ŞEHİR UNIVERSITY

- We've managed to retrieve pages that match the queries.
- The order in which they are returned is simply the order in which they were crawled.
- In a large set of pages, you would be stuck sifting through *a lot of irrelevant content* for any mention of each of the query terms in order to find the pages that are **really related to your search.**

Score Computation

İSTANBUL ŞEHİR UNIVERSITY

```
def getscoredlist(self, results, words):
        totalscores = dict([(url, 0) for url in results])

        # word frequency scoring
        weights = [(1.0, self.frequencyscore(results)),
                   (1.0, self.locationscore(results)),
                   (1.0, self.inboundlinkscore(results))]

        for (weight,scores) in weights:
            for url in totalscores:
                totalscores[url] += weight*scores.get(url, 0)

        return totalscores
```

Normalization

İSTANBUL ŞEHİR UNIVERSITY

- In order to compare the results from different scoring methods, we need a way to normalize them.
- The normalization function: each score is scaled according to how close it is to the best result, which will always have a score of 1.

Ranking

İSTANBUL ŞEHİR UNIVERSITY

- Content-based Ranking
- Inbound-link Ranking

Ranking measures

İSTANBUL ŞEHİR UNIVERSITY

- **Word frequency:** the number of times the words in the query appear in the document can help determine how relevant the document is.
- **Document location:** the main subject of a document will probably appear near the beginning of the document.
- **Word distance:** if there are multiple words in the query, they should appear close together in the document.

Ranking

İSTANBUL ŞEHİR UNIVERSITY

```
import mysearchengine

dbtables = {'urllist': 'urllist.db', 'wordlocation':'wordlocation.db',
        'link':'link.db', 'linkwords':'linkwords.db', 'pagerank':'pagerank.db'}

searcher = mysearchengine.searcher(dbtables)

searcher.query('computer science')
```

Word Frequency

İSTANBUL ŞEHİR UNIVERSITY

- The word frequency metric scores a page based on how many times the words in the query appear on that page.
- Search for "python"

  - a page about *Python (or pythons) with many mentions of the word python.*

  vs.

  - *a page about a musician that has a pet python.*

Word Frequency

İSTANBUL ŞEHİR UNIVERSITY

```
def frequencyscore(self, results):
    counts = {}
    for url in results:
        score = 1
        for wordlocations in results[url]:
            score *= len(wordlocations)
        counts[url] = score
    return self.normalizescores(counts, smallIsBetter=False)
```

Word Frequency

İSTANBUL ŞEHİR UNIVERSITY

- change the weights line in getscoredlist to this:

weights=[(1.0, self.frequencyscore(rows))]

```
import mysearchengine
dbtables = {'urllist': 'urllist.db', 'wordlocation':'wordlocation.db',
      'link':'link.db', 'linkwords':'linkwords.db', 'pagerank':'pagerank.db'}
searcher = mysearchengine.searcher(dbtables)
searcher.query('computer science')
```

Document Location

İSTANBUL ŞEHİR UNIVERSITY

- Search term's location in the page.
- If a page is relevant to the search term, it will appear closer to the top of the page, perhaps even in the title.
- Score results higher if the query term appears early in the document.
- wordlocation **table: the locations of the words were recorded.**

Document Location

İSTANBUL ŞEHİR UNIVERSITY

```
def locationscore(self, results):
    locations=dict([(url, 1000000) for url in results])
    for url in results:
        score = 0
        for wordlocations in results[url]:
            score += min(wordlocations)
        locations[url] = score
    return self.normalizescores(locations, smallIsBetter=True)
```

Document Location

İSTANBUL ŞEHİR UNIVERSITY

- change the weights line in getscoredlist to this:

weights=[(1.0, self.locationscore(rows))]

```
import mysearchengine
dbtables = {'urllist': 'urllist.db', 'wordlocation':'wordlocation.db',
      'link':'link.db', 'linkwords':'linkwords.db', 'pagerank':'pagerank.db'}
searcher = mysearchengine.searcher(dbtables)
searcher.query('computer science')
```

Word Distance

İSTANBUL ŞEHİR UNIVERSITY

- When a query contains multiple words, seek results in which the words in the query are close to each other in the page.
- Multiple-word queries, seek pages that conceptually relates the different words.

Word Distance

İSTANBUL ŞEHİR UNIVERSITY

Left as a take-home exercise!

Link information

İSTANBUL ŞEHİR UNIVERSITY

- Used scoring metrics based on the content of the page.
- How about information that others have provided about the page, who linked to the page, what they said about it?

Pages created by spammers are less likely to be linked than pages with real content.

Link information

İSTANBUL ŞEHİR UNIVERSITY

- The link table has the URLs for the source and target of every link that it has encountered.
- The linkwords table connects the words with the links.

Simple Count

İSTANBUL ŞEHİR UNIVERSITY

- Count inbound links on each page and use the total number of links as a metric for the page.
- The scoring function based on this count:

```
def inboundlinkscore(self, results):
    inboundcount=dict([(url, len(self.link[url]))
                       for url in results if url in self.link])
    return self.normalizescores(inboundcount)
```

Simple Count

İSTANBUL ŞEHİR UNIVERSITY

- Using this metric by itself will simply return all the pages containing the search terms,

  "*ranked solely on how many inbound links they have.*"

- We need to combine the inbound-links metric with one of the content/relevance metrics we saw earlier.

What if?

İSTANBUL ŞEHİR UNIVERSITY

- Someone can easily set up several sites pointing to a page whose score they want to increase.

What about!

İSTANBUL ŞEHİR UNIVERSITY

- What about results that have attracted the attention of very **popular** sites?
- How to make *links from popular pages worth more* in calculating rankings?