Web Crawling

Given a web page link, the Perl script in this folder crawls (downloads) the main page of the website and on all other documents directly linked from the page. Displays the vocabulary and frequency of each word in them. Also computes in how many different documents in this small collection each word occurs.

```
#
# VERSION HISTORY:
# Rajendra Banjade 9/26/2012
#

use LWP::Simple;                # utility package (use get() to fetch content
of url etc).
use CAM::PDF;                   # to process pdf files.
use CAM::PDF::PageText;         # for pdf to text conversion.

$baseUrl = "http://www..../";  # webpage link

$homePageContent = get($baseUrl) || die "\nCouldn't fetch $baseUrl\n";  #
first fetch the home page content.

my @links = $homePageContent =~ /href="(.*?)"/gi; # get all (g) the url's the
page linked to.
my $crawledPageCount = 0;                         # counter for pages crawled
(not all links be crawled - duplication, anchor links etc).
my $wordCount = 0;                                # total number of words
found.
my %wordDocFrequencyTable = ();                   # word document frequency
table.
my %wordFrequencyTable = ();                      # word frequency table.
my %uniqueLinkTable = ();                         # unique links crawled.

# Parses the page content, and counts word frequency & document frequency for
each word.
sub processData{
    my ($pageData, $url) = @_;
    my %tempHash = ();                          # Store words whose document
frequency has been updated.

    $pageData =~ s/\s+/ /g;                     # replace any sequence of
whitespaces by a single whitespace (for easy split)
    $pageData =~ s/<.+?>//g;                    # remove valid html tags
(opening | closing), do shortest match (?).
    my @tokens = split /\s+/, $pageData;        # split by whitespace
    my @words = grep /^[A-Za-z0-9]+$/, @tokens; # get the valid word list
    $wordCount += @words;

    #update the word frequency, and document frequency (if needed) tables.
    foreach $word (@words) {                     # Case sensitive, match.
        $wordFrequencyTable{$word}++;
        if (! (exists $tempHash{$word})){
            $wordDocFrequencyTable{$word}++;     # update in the document
```

```
frequency table.
            $tempHash{$word} = 1;                  # Flag: Document frequency
for this word has been updated.
        }
    }
}

# First process the course home page.
&processData ($homePageContent, $baseUrl);
$uniqueLinkTable{$baseUrl} = 1;                    # mark home page as
processed. Avoid potential duplications.
$crawledPageCount++;

# for each link, either discard or get data and process.
foreach $link (@links) {
    next unless (!($link =~ /#|mailto:/));          # if anchor link or email
link, just skip.
    next unless (!($link =~ /\.ppt/));              # .ppt file, skip it (for
now).

    if (!($link =~ /^http\:\/\//)) {                # if relative path,
change to absolute. Assume no link to secure (https) pages
        $link = $baseUrl.$link;
    }

    next unless (! (exists $uniqueLinkTable{$link}));   # Avoid redundant
processing (i.e. Already processed link, skip it)

    # Handle different type of files differently (.pdf,.html etc).
    if ($link =~ /\.pdf/){
        my $tempPdfFile = "temp.pdf";                   # temporary pdf dump
file.
        my $tempTextFile = "temp.txt";                  # temporary file for
pdf to .txt
        $pageContent = get ($link) || next;             # fetch the pdf
content, otherwise skip.
        next unless (open (TEMPFILE, ">$tempPdfFile")); # if can't create
temp file, just skip that pdf.
        binmode(TEMPFILE);                              # pdf must be saved
as binary file !!!
        print TEMPFILE $pageContent;
        close (TEMPFILE);
        my $pdf = CAM::PDF->new($tempPdfFile);           # read pdf file.
        my $pageCount = $pdf->numPages();                # get number of
pages
        next unless (open (TEMPFILE, ">$tempTextFile")); # if can't create
temp file, just skip that pdf.
        # Convert each pdf page to txt and Append to.
        foreach my $pageNum (1..$pageCount) {
            my $page = $pdf->getPageContentTree($pageNum);
            print TEMPFILE CAM::PDF::PageText->render($page);
        }
        close (TEMPFILE);
        next unless (open (TEMPFILE, "<$tempTextFile")); # if can't read
converted text, just skip that pdf.
        my @content = <TEMPFILE>;
        $pageContent = join " ",@content;   # change to single string (space
```

```
separated for each line).
                                                    # Just to process like other HTML
page's content.
        close (TEMPFILE);
        #remove temp (.pdf and .txt) files.
        unlink($tempPdfFile);
        unlink($tempTextFile);
    } else {
        $pageContent = get ($link) || next;  # otherwise, assume the page is
html and fetch it.
    }
    &processData ($pageContent, $link);      # Process the page data.
    $uniqueLinkTable{$link} = 1;             # mark as processed.
    $crawledPageCount++;
}

# print crawled page urls (some converted to absolute though they were
originally relative to the course home page)
print ("\n ======== Crawled page urls ====== \n");
my $count = 0;
foreach $url (sort keys %uniqueLinkTable ){
    $count++;
    print ("\n$count> $url");
}

# sort in the alphabetical order and display word frequency table.
print ("\n\n ===  Word => Frequency table === \n");
foreach $word (sort keys %wordFrequencyTable ){
    print ("\n$word ----> $wordFrequencyTable{$word}");
}

# Sort words in the alphabetical order and print document frequency table.
print ("\n ===  Word => Document frequency table === \n");
foreach $word (sort keys %wordDocFrequencyTable ){
    print ("\n$word ----> $wordDocFrequencyTable{$word}");
}

# print some statistics
print ("\n\n Total pages crawled: $crawledPageCount , Total word count:
$wordCount , Unique word count: ".keys(%wordFrequencyTable). "\n\n");
```

Src/WebCrawler.pl

NOTES:

- ⇨ To process the pdf file, external modules CAM::PDF and CAM::PDF::PageText (both from CPAN) has been used in this program. They are useful while converting the pdf file to plain text and then we can update the word and document frequency. Similarly, we can use some external modules to handle power point slides as well. However, they are now skipped as the way we process pdf file has given some idea of crawling non HTML pages.

- ⇨ Vocabulary contains case sensitive words. It increases the vocabulary size but sometimes helpful determining the meaning that word carries. For example, Ram – a name, RAM – Random Access Memory.
- ⇨ We are not saving the downloaded pages. We just tried how to download the page and simply process on the fly. It's kind of odd. However, in the next phase, we will save and preprocess.

To install CAM::PDF module,

- ⇨ Go to CPAN client (in windows 7, please run it as administrator).
- ⇨ Issue a command, install CAM::PDF

To run the program,

- ⇨ Go to the <your folder>/src
- ⇨ Specify a web page in the code, which is at the beginning of the code.
- ⇨ Issue command, perl WebCrawler.pl > output.txt
  Since the output is very large, it would be a good idea to redirect to some file (as shown above) and see.

The program basically,

1. It fetches the content from the page (home page) specified by the user.
2. Gets the URLs of the pages directly referenced from that page.
3. For each URL (except link to .ppt file, anchors, mailto: links) including the course home page,if the link is relative, appends in the base URL to make it absolute one and gets the page content.
4. If the link was to pdf file, saves it to a pdf file (binary format) and converts to text file using external library CAM::PDF and reads the whole text file at once as a single web page.
5. Removes the html tags
6. Replaces multiple spaces by single space
7. Splits the string by space character and keep the valid words only (now the valid word means the word containing only alpha numeric characters).
8. Counts the word frequency keeping them in the hash of word as key, and frequency as value. If that word has not been encountered already, it updates the document frequency as well.
9. Displays the word – frequency, and document frequency on the sorted order of words.

Sample output:

The output would look like,

```
======== Crawled page urls ======
```

```
1> http://tartarus.org/~martin/PorterStemmer/
2> <other web page links are removed from here..>
3> ..
4>..

 ===  Word => Frequency table ===

0 ----> 12
02 ----> 1
03 ----> 1
…..
ACM ----> 2
ANSI ----> 5
ANTS ----> 1
AT ----> 1
Ac ----> 1
Access ----> 3
Accessed ----> 3
According ----> 1
Acknowledgements ----> 1
Advanced ----> 2
Affairs ----> 1
…..
===  Word => Document frequency table ===

0 ----> 1
02 ----> 1
03 ----> 1
04 ----> 1
05 ----> 1
06 ----> 1
….
A ----> 7
ACM ----> 1
ANSI ----> 1
ANTS ----> 1
AT ----> 1
Ac ----> 1
Access ----> 1
Accessed ----> 1
According ----> 1
Acknowledgements ----> 1
Advanced ----> 1
…
zakona ----> 1
zavisyaschie ----> 1
zu ----> 2

 Total pages crawled: 9 , Total word count: 10141 , Unique word count: 2570
```