

Name: Dien Nguyen  
UID: 804332365

PIC16 Final Project (Search Engine)

## Search Engine, aka AltaVista v.0.1

Have you ever had to find stuff on the internet, but only wanted to look through certain websites? Of course you have. Here's one way to solve that problem, in Python 2.  
Jokes...

Problem Description (from  
[http://www.math.ucla.edu/~rombach/PIC16\\_project2.pdf](http://www.math.ucla.edu/~rombach/PIC16_project2.pdf)):

Create a mini search engine. This engine uses a list of urls as its “internet”.

- Your engine needs to create a network based on links between the pages which will be in the source code of any page)
- The engine understands the meaning of quotation marks.
- The engine does pagerank on the sub-network of the pages that contain the search term from the user, and then lists the first few, together with a small snippet of the text on the page. This should be a snippet that contains the highest concentration of the search term words.
- For starters, you may want to create some pages to form your own little internet as a test case.
- Your engine recognizes typos, and says “Did you mean...?” The suggestion it gives will be the word in its dictionary (for example: [mieliestronk.com/corncob\\_lowercase.txt](http://mieliestronk.com/corncob_lowercase.txt)), that is closest to the search term.

## Description of the SearchEngine class:

Let's call an instance of the SearchEngine class SearchEngine\_object

1. SearchEngine initializes with a set of urls and a list of words as a "dictionary".
2. It then generates a network of links between the pages.
3. It also cache the pages, both the source code and the text, so it doesn't have to keep reading the page every time it wants to do something with the page.
  - a. The cached source codes are used for generating the network and for getting titles of each website.
  - b. The cached texts are used to query the SearchEngine and to get the snippets.
4. When the user wants to find something in their mini-"internet", he/she can query the SearchEngine instance.
  - a. The engine tries to catch misspelled words and asks the user if he/she wants to revise his/her query, while giving him/her a couple of likely options.
  - b. The engine prints out the results, ranked by the PageRank algorithm and grouped by relevancy. Each result consists of the url, the title, and a snippet of the page. The return value is a list of urls in the same order.
5. The user can also:
  - a. Update the SearchEngine\_object's dictionary using the function `update_dict(new_dict)`.
  - b. Refresh the SearchEngine\_object's cache by using the function `refresh_cache()`. This is useful when the pages in the "internet" changes.

Libraries Needed: urllib2, re, numpy

## Pseudocode for each function:

function update\_dict(new\_dict)

- Set the local dictionary to new\_dict

function refresh\_cache()

- For each url in the “internet”

  - Read each page’s source code

  - Save it in a list, \_cached\_pages

  - Convert the page’s source code to readable text, using \_strip\_html()

  - Save it in another list, \_clean\_cached\_pages

- Generate the network

function \_strip\_html(txt)

- Find each html tag in string txt using Regular Expression and delete it

- Replace each whitespace in txt with a space (‘ ’)

function \_generate\_network()

- Initiate a numpy array of zeros, representing the number of links between pages

- For each page in the “internet”

  - Get a list of urls in each page’s source code, again using RegEx

  - Change any relative url to its full url form

  - For each url in this new list

    - If the url is in our “internet”

      - Increment a connection in the right indices in our network

- Return network

function query(input\_query, num\_results)

- Split the input\_query into quoted and unquoted components using the function

- \_parse\_search\_query()

- Checks for spelling error using function \_find\_closest()

  - If there is, asks if user wants to change their query

- Get a list of ranked urls matching the query using the function \_match\_and\_find\_urls()

- For each url in this list

  - Obtain and print the output (url, title, snippet(with \_snippet())) for each url

- Returns the list of ranked urls

function \_parse\_search\_query(query)

- Find the quoted component using RegEx

- Find the unquoted component by removing quotation marks from query and split it by whitespace

- Return lists of quoted and unquoted components

function \_find\_closest(misspelled)

    Get a list of edit distances using helper function \_dist()

    Get the minimum edit distance

    Return a list of words with the minimum edit distance

function \_dist(word1, word2)

    Initialize a matrix of edit distances from word1 to word2, each index (i,j) represent the minimum edit distance from word1[ : i ] to word2[ : j ]

    For each row index matrix

        For each column index in matrix

            If row index == 0

                matrix[row index, col index] = col

            Else if col index == 0

                matrix[row index, col index] = row

            Else if last characters of word1 and word2 are the same

                matrix[row index, col index] = matrix[row index-1, col index-1]

            Else

                Take the min of the above, left, and top left elements + 1

    Return matrix[length of word1, length of word2]

function \_match\_and\_rank\_url(quoted, unquoted, num\_results)

    To get the ranks based on relevancy,

    For each page in "internet"

        Count number of number of unique matches for quoted and unquoted to get the ranks based on relevancy

        If there are any matches, add the index of the page to a list of indices

    Construct a subnetwork based on the set of indices using function \_subnetwork()

    Run PageRank on the subnetwork using function \_pagerank()

    Until we have at least n\_results pages to return

        Rank the pages based on PageRank and group the them by relevancy scores

    Return only the top num\_results pages

function \_subnetwork(indices)

    List comprehension for each element in indices twice, columns and rows

function \_pagerank(N,p)

    Cast the network N as a numpy array with float as the type of each element

    Normalize N+p, where p is a small constant, call this N

    Obtain eigenvectors and eigenvalues of N

    Take absolute value of all eigenvalues

    Find maximum eigenvalue, 1, and get its corresponding eigenvector, the scores

    Return a list of corresponding ranks of the scores

function \_normalize(net)

Divide each column in net by its own sum and return new network

function \_snippet(index, quoted, unquoted)

Obtain the page from clean cache at index

If the page is too small

Split the page, by '\n', into 2 lines and return it

#else

Use a moving window on the page and find the window that have the most matches

Split the window, by '\n', into 3 lines and return it

## Demo on an Artificially Constructed Network

Network: <http://pic.ucla.edu/~dpnguyen/pic16/> (might be down temporarily)

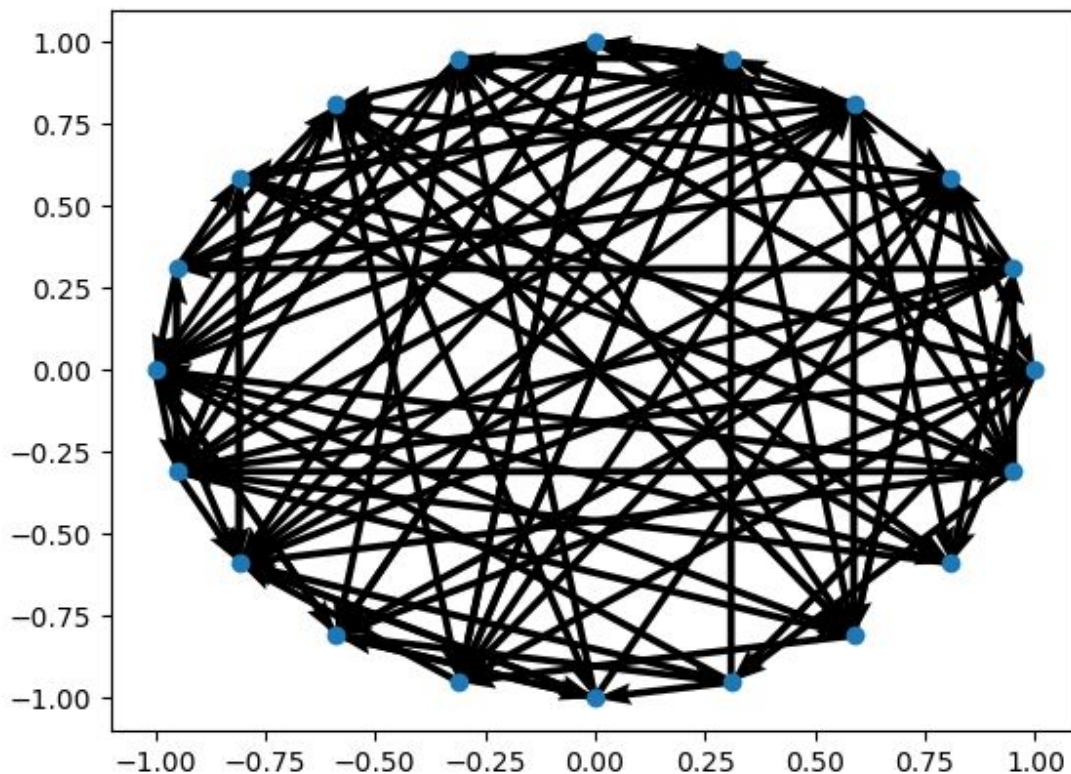
Dictionary: [http://mieliestronk.com/corncob\\_lowercase.txt](http://mieliestronk.com/corncob_lowercase.txt)

Code:

```
exec(open("./SearchEngine.py").read())
english_dict = urllib2.urlopen( 'http://mieliestronk.com/corncob_lowercase.txt').read().split()
# personal fake internet
urls = ""
for i in range(20):
    urls += 'http://pic.ucla.edu/~dpnguyen/pic16/page%d.html ' %i

cool_engine = SearchEngine(urls,english_dict)
```

Network Visualization (20 nodes):



Sample Query Output for cool\_engine.query('coffee "money" beans taes lef price', 5):

['clef', 'lea', 'leaf', 'led', 'lee', 'left', 'leg', 'let', 'ref']

It seems that you misspelled "lef". Some suggestions are displayed above?

Do you want to change your input for this word (y/n)? y

To what? >? leaf

['tabs', 'tags', 'takes', 'tales', 'tames', 'tans', 'tapes', 'taps', 'tares', 'tars', 'taxes', 'tees', 'ties', 'toes', 'tues']

It seems that you misspelled "taes". Some suggestions are displayed above?

Do you want to change your input for this word (y/n)? y

To what? >? bagteas

['bagels', 'baiters', 'bates', 'battens', 'batters']

It seems that you misspelled "bagteas". Some suggestions are displayed above?

Do you want to change your input for this word (y/n)? n

<http://pic.ucla.edu/~dpnguyen/pic16/page0.html>

Wikipedia Finance

Economic history [show] v t e Finance is a field that deals with the study of investments. It includes the dynamics of assets and liabilities over time under conditions of different degrees of uncertainty and risk.

Finance can also be defined as the science of money management. Finance aims to price

<http://pic.ucla.edu/~dpnguyen/pic16/page10.html>

Wikipedia Communist state

ne Transaction. p. 21. ISBN 978-0202362281. "Contrary to Western usage, these countries describe themselves as 'Socialist'

(not 'Communist'). The second stage (Marx's 'higher phase'), or 'Communism' is to be marked by an age of plenty, distribution according to needs (not work), the absence of money

<http://pic.ucla.edu/~dpnguyen/pic16/page5.html>

Wikipedia Coffee

ins even though actual coffee beans are not added to it.[112][113] Instant coffee Main article: Instant coffee

Instant coffee A number of products are sold for the convenience of consumers who do not want to prepare their own coffee or who do not have access to coffeemaking equipment. Instant coffee

<http://pic.ucla.edu/~dpnguyen/pic16/page9.html>

Wikipedia Privately held company

ot Corp., Deloitte Touche Tohmatsu (one of the members of the Big Four accounting firms), Hearst Corporation, Cox Enterprises, S. C. Johnson, McWane, Carlson Companies, and Mars are among the largest privately held companies

in the United States. KPMG, the UK accounting firms Ernst & Young and Price

<http://pic.ucla.edu/~dpnguyen/pic16/page12.html>

Wikipedia Ethiopia

. [157][158] Coffee remains its most important export product, and with new trademark deals around the world (including recent deals with Starbucks) the country plans to increase its revenue from coffee. [159] Most regard Ethiopia's large water resources and potential as its "white oil" and its coffee

Return Value

['<http://pic.ucla.edu/~dpnguyen/pic16/page0.html>',  
'<http://pic.ucla.edu/~dpnguyen/pic16/page10.html>',  
'<http://pic.ucla.edu/~dpnguyen/pic16/page5.html>',  
'<http://pic.ucla.edu/~dpnguyen/pic16/page9.html>',  
'<http://pic.ucla.edu/~dpnguyen/pic16/page12.html>']