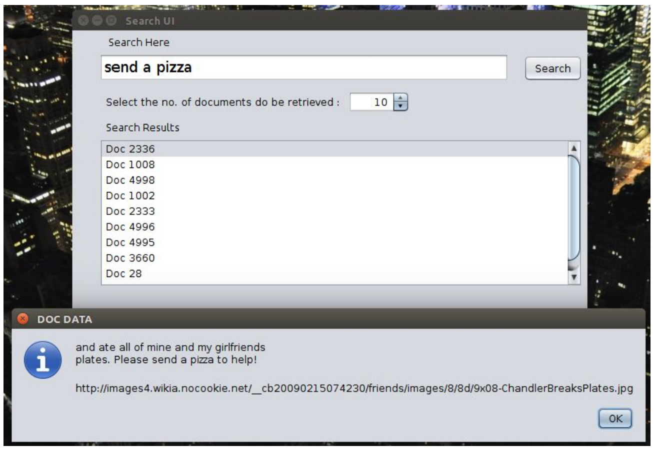
Design Document on Building a Search Engine

Submitted on: - 27th September 2016 **Submitted to -Dr. Aruna Malapati**

Submitted by: -

Soamya Agrawal	2014A7PS185H
Ayushi Behl	2014A7PS145H
Tejeshwar Reddy	2014A8PS492H
Ruthvi Reddy	2014A7PS040H

About the search engine:-
The project implements a Probabilistic Information Retrieval System using JAVA. For easy user- document interaction GUI has been implemented, which shows the output (relevant documents) of the query in the form of a list. The user can right-click on a list item and select 'show' to view the content of the document. The content of the document will be shown in a message dialog box. The entire GUI is made by using the swing library in NetBeans.



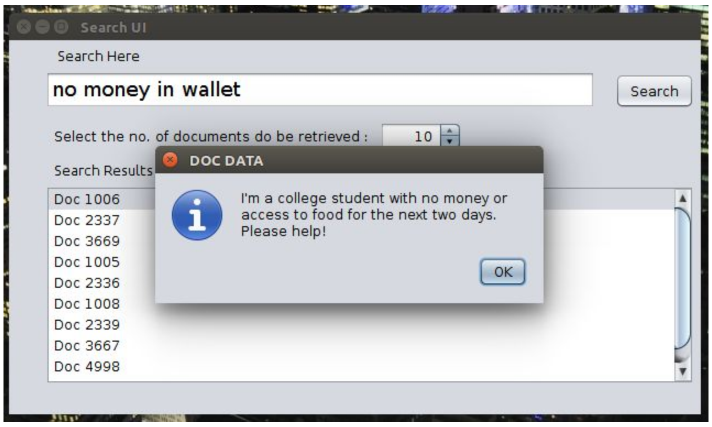
send a pizza

10

Doc 2336
Doc 1008
Doc 4998
Doc 1002
Doc 2333
Doc 4996
Doc 4995
Doc 3660
Doc 28

and ate all of mine and my girlfriends
plates. Please send a pizza to help!

http://images4.wikia.nocookie.net/__cb20090215074230/friends/images/8/8d/9x08-ChandlerBreaksPlates.jpg

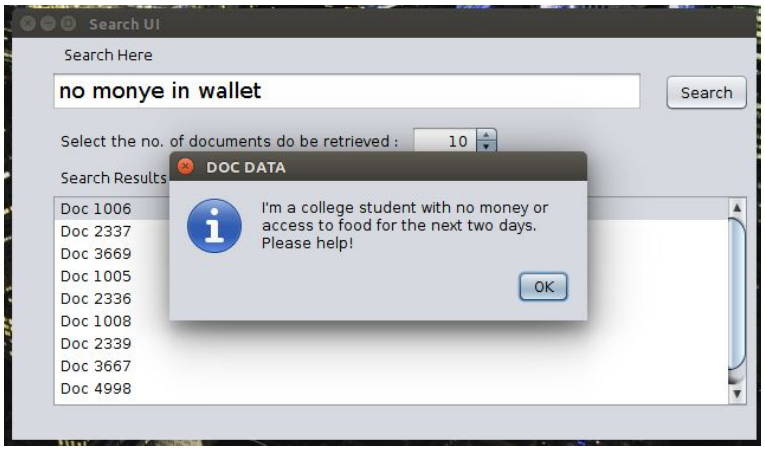


no money in wallet

10

Doc 1006
Doc 2337
Doc 3669
Doc 1005
Doc 2336
Doc 1008
Doc 2339
Doc 3667
Doc 4998

I'm a college student with no money or
access to food for the next two days.
Please help!



no monye in wallet

10

Doc 1006
Doc 2337
Doc 3669
Doc 1005
Doc 2336
Doc 1008
Doc 2339
Doc 3667
Doc 4998

I'm a college student with no money or
access to food for the next two days.
Please help!

Methodology:-

Tokenization:- Built-in classes namely PTBTokenizer,CoreLabelTokenFactory and CoreLabel available in edu.stanford.nlp.process package was used for tokenizing the query and the documents.

Corpus:- https://snap.stanford.edu/data/web-RedditPizzaRequests.html using this link the corpus for the assignment was obtained. A JSON file was retrieved after unzipping the folder available in the above link. To extract the documents from this file, we used the JSONArray class available in the org.json package.

Stemming:-After Tokenization the tokens were stemmed using the Porter's Algorithm.

Its implementation was done using a class Stemmer.

Metaphone:-This is a phonetic algorithm, which does a better job than the Soundex algorithm
of matching words and names which sound similar. It is implemented by defining a method
named encode() which takes the stemmed tokens as its argument.

Term Frequency Calculation:-All the tokenized, stemmed and metaphoned distinct words form the terms of dictionary of individual documents. The dictionary is implemented using the TreeMap data structure, which provides an efficient means of storing key(term)/value(term frequency) pairs in sorted order and allows rapid retrieval.

ArrayList Implementation:-All the TreeMaps of individual documents are listed together using data structure ArrayList.It is used to support dynamic arrays that can grow as needed.

Total Number of Documents Calculation:-Using the size() method of ArrayList, the total number of documents were retrieved.

Document Frequency Calculation:-A TreeMap is used having key(all the distinct terms present in ArrayList)/value(their document frequency) pairs.

Rank Calculation:-To obtain the rank of each document, the formula-

$$P(R=1|D) \overset{rank}{=} \frac{P(R=1|D)}{P(R=0|D)} = \frac{P(D|R=1)\,P(R=1)}{P(D|R=0)\,P(R=0)}$$

where,

w: term;

D: dictionary comprising all the distinct terms;

Q: query as input by user;

N: total number of documents available in the corpus;

N_w: document frequency of the term

To display the ranked documents in descending order (in order of relevance to the query) a
sortByComparator() method was defined.