Analysis of Authority and Hub Algorithm

Question) what effect does changing the root set size has on quality (relevance) of results? Why?

Answer: If we decrease the root set size then we will have less number of documents to do our Auth-Hub computation on, hence decreasing root set size will decrease the relevant documents. However, if we increase the root set size (say 20) then it would affect the relevance marginally, but would not decrease the quality of results.

Since when root set size is increased then documents who have low tf –idf similarity values end up getting high Authority or Hub value (because of the inlinks and outlinks), increasing the root set size gives a chance to new documents to end up as high Authority page or hub page for a given query which would have been never ended up in top ten had we taken the small root-set.

For example, when we increased the root set size to 20 for query word "Employee Benefits" then we get the document number 4560 as a high authority page. Also, for query word "transcripts" we observed document number 22936 comes up in results, which is a very good page about transcripts.

Question) what effect does changing the root set size have on time taken? Why?

Answer:

Size of Root Set (k)	Time taken in Mili-Seconds (Query=**Admissions**)	Time taken in Milli Seconds (Query=**Campus Tour**)
10	612	1244
20	953	1289
30	1071	1352
40	1412	1392
100	1954	1854
400	29987	35545

As we can observe, when we increase the size of root set (k) for query words then time taken increase but not drastically until we reach a certain level (when k becomes 400). At k=400, the effect on time taken is drastic since when root set increases then our base set grows since more number of documents in root set means more number of links/citations and thus a bigger base set and hence more number of computations.

Question) Compare the time taken by various phases of the algorithm.

Answer:

Query	**Time taken to get the root set using TF-IDF (in Milliseconds)**	**Time taken to get the base set(in Milliseconds)**	**Time taken to compute Authority and Hubs(in Milliseconds)**	**Time taken to Sort to Authority and Hub vectors (in Milliseconds)**
Employee benefits	5630	13	21	4

Transcripts	870	6	2	1
SRC	560	11	3	2
Parking Decal	601	5	6	1
Campus Tour	607	7	15	2
Admissions	402	2	4	0

During Authority and Hub computation, time is being spent to find the root set using TF-IDF method and finding base set from the root set since both computations goes over substantial number of documents to find the results. The other two steps takes less time in comparison to the first two since matrices converges very fast.

Question : **Which results do you think are more relevant - TF/IDF, Authorities, Hubs or Pagerank? Why? Support your answer with examples or statistics.**

Answer:
TF-IDF vs Auth/HUB: When we compare the results which are given by TF-IDF method and by Authority & Hub method then we would observe that the results given by TF-IDF are more relevant since TF-IDF method fetches those documents which have more occurrence of query words and importance of those query word among whole document corpus (IDF), where as Authority and Hub focuses on links /citations of a given a document which dilute the root set with some irrelevant documents. Above discussed approach will be explained by using an example below.

Example: For query "Admissions" TF_IDF returns page as most relevant is www.asu.edu/admissions/contact/index.html whereas, the top authority page is www.asu.edu/admissions/whychooseasu/location.html. When we observed both web pages then, the previous page (returned by TF-IDF) was more relevant and appropriate for this query.

TF-IDF versus PageRank: We can observe the difference by looking at the results returned by both approaches. However, we are using equation *w * (PageRank) + (1-w)* (Vector Space Similarity)* so we can say that vector similarity and PageRank both contributing to resultant similarity depending on number 'w'. If 'w' is kept high then PageRank will have more influence on resultant similarity whereas if 'w' is kept low then vector similar will play a big role in deciding the resultant similarity.
Also, we observed that PageRank is not diluting results if 'w' is kept low, which is a vital point in deciding the final similarity.

Example: For query "Transcripts" when we set the value to 0.4 then we got document www.asu.edu/registrar/transcripts/index.html as the top page and we got the same top page when we changed w to '0.6', but, top page changed www.asu.edu/secure/archives.html when we changed the value of 'w' from 0.6 to 0.9.
On close observation, previous page was more relevant for the query word.

Analysis of Page Rank Algorithm

Question) What effect does varying w have on the relevance of the results? Why?
Answer: We have computed the importance using these formulae.

*w * (PageRank) + (1-w)* (Vector Space Similarity)*

If we take high value of 'w' then more weightage will be given to page rank value of documents rather than TF-IDF values of page and hence User will get more pages which have high number of links rather than the pages that contain the more occurrence of query terms and thus relevance will decrease.

On the other side, if we decrease the value of 'w' then more weightage will be given to TF-IDF values of pages, hence user will get those pages which contain the more occurrences of query terms and thus relevance will increase.

For example: For query "parking decal" when we set 'w' as 0.2 and 0.4 we got document id 649 as the top document and 2406 as second best but when we change 'w' as 0.6 we got the document id 2406 as the top page and 649 as the second best page. On closer observation document Id 649 was more relevant to query in comparison to document id 2406. Which clearly indicates the affect of varying 'w' on the relevance of results for a given query.

Question) what effect does varying c have on the relevance of the results? Time taken? Why?

Answer: If we take high value of 'c' then as per random surfer model, user will go to those pages, which are being linked by a current page (on which user is on). On the contrary, if we take low value of 'c' then it means, user is supposed to randomly go to other pages, which are not being pointed by the current page the user is on.

Also, when 'c' increases then it takes very long to converge hence time taken increases. But, it does not affect relevance because the formula (M*=c(M+Z)+ (1-c)K) suggests that c is probability that the user will go to those pages which are being linked by current pages. Hence, when 'c' decrease then it implies that user can go those pages, which are not linked by current pages. Thus, changing the value of 'c' would not affect relevance of results for a given query.

Question) did your PageRank computation converges? How many iterations did it take? How much memory did it take?

Answer: No, computation did not converge until I make an approximation of decimal digits.

However, computation converged after 9 iterations once I approximated the decimal digits. I checked the previous vector with the new vector and approximated the 4^{th} digit in the decimal point so I can reduce the number of iterations. Approximation in the 4^{th} digit did not make any significant difference in the final top rankings of documents. Approximately, it takes 2*25054*8(i.e. 2 arrays of size 25054 and each array is of double type i.e. 0.3822 MB).

However, when I tried using Runtime class in java (runtime.totalMemory()-runtime.freeMemory()), this indicated that approximately 400MB is being used by the program, it also includes the java runtime environment variables.

Question) which document had the highest PageRank? Does that make sense?

Answer: Document '9048' has the highest PageRank. Since there are so many documents which have links to this document i.e. document '9048' has 7250 citations, which makes it a very important document of the corpus hence it makes sense.

Question) is there any correlation between high authority values and high PageRank values? What about hub values and PageRank values? Does this make sense?

Answer: Yes, we can argue that there is a correlation. Document with high authority values have will have high page rank values. High value of authority means that there so many documents which are pointing to a page or documents which have high number of citations. If there are too many documents pointing to a page then its page rank will be high as well. For example, Document 9048 has high page rank value as well as high authority value (if we do Hub-Auth computation on entire corpus).

Similarly, hub and page rank values are related with each other as well. For example, A pure hub will be a node which is not being pointed by any other node i.e. number of citations are zero, it also means that page rank for those documents will be low. For example, documents 925,958,1068,1069 have zero number of citations and also page rank value as zero.

K-means Clustering Algorithm Analysis

Cluster Summary:

Approach/Algorithm:

1. Once mean vectors/Centroid Vector converged to a value where all documents settled in their respective clusters i.e. after the iteration when no documents is going to a new cluster.
2. For every mean Vector/ Centroid Vector (there will be 'k' mean vectors/ Centroid vectors for 'k' clusters), find top six values in mean Vectors/ Centroid vectors and check their corresponding terms.
3. Terms picked in step 2 will represent the respective cluster summary.

Question) Pick any two queries from the set given below. Change the value of 'k' between 3 and 10. What do you observe? Why?

Answer: We picked admissions and languages as query terms and found that if we increase the value of k then K means clustering will take more time to converge since there will be more number of iterations. Also, We observed that clusters would be tighter as value of k increases.

Question) how does execution time change?

Answer: Execution time will increase as the number of iteration will also increase. We can observe from table 1 and 2, that as the value ok 'k' increases time taken to iterate will also increase. This is the due to the fact that there will be more number of computations when same number of documents has to be settled in more number of clusters.

For query "**admissions**"

Value of k/Time taken	Time taken to fetch TFIDF Documents (in Milli seconds)	Time taken to compute document-term vector (in Milli seconds)	Time taken to iterate (in Milli seconds)
k=4	5187	20089	6456
k=5	5169	20227	8024
k=7	5796	22175	11559
k=8	5912	22189	12907

Table 1: Time taken at respective steps for query "admissions"

For query "**languages**"

Value of k/Time taken	Time taken to fetch TFIDF Documents (in Milli seconds)	Time taken to compute document-term vector (in Milli seconds)	Time taken to iterate (in Milli seconds)
k=4	5176	20187	4999
k=5	5393	21160	6036
k=7	5873	20384	11153
k=8	5331	20916	12753

Table 2: Time taken at respective steps for query "languages"

For example, for query "languages" we can observe that time taken to iterate increases as value of 'k' increases (from table 2), this is also true for query "admissions". Since, we need to re-compute mean-vector/Centroid-vector for each cluster after every iteration hence, as value of 'k' increases number of mean- Vector/Centroid-Vector also increases and thus more time is required to re-compute the mean-vector/Centroid-Vector for each cluster. Also, there will be more number of comparisons of documents to the centroid vector in each iteration.

Question) How does the similarity of the document to the centroid of the cluster change?

Answer: As value of 'k' increases the intra cluster distance in a cluster decrease as clusters will become more compact and tight. And, all those documents that are less similar to a cluster will leave the cluster and settle down in a new cluster; also, existing documents in the cluster will become more similar to the mean-vector/ Centroid-Vector or the centroid.

For example:

For query "admissions" when we set k=4, document id 939 is mapped to cluster 1 and it has similarity value as 0.88026764 with the centroid vector, when we increased the value of k to 5, we observed that its similarity with the centroid vector increases to 0.93965610 and when k is set to 7 then its similarity increases to 0.946897767. This observation hints towards a fact that as value of 'k' increases then the similar document will become more similar.

Hence, we can conclude that as value of 'k' increases then we will get tighter cluster.

Question) How did the value of k affect the clustering? Justify with a couple of examples.

Answer: We have taken "admissions" as a query and observed the following.

For, total Docs=10 and k=4

www.asu.edu/admissions/international/faq.html (document id 963) -> **Cluster 0**

www.asu.edu/admissions/contact/transfer.html (document id 941) -> **Cluster 1**
www.asu.edu/admissions/whychooseasu/index.html (document id 1081)->**Cluster 1**
www.asu.edu/admissions/contact/freshman.html (document id 937) -> **Cluster 1**
www.asu.edu/admissions/contact/familymember.html(document id 936)->**Cluster 1**
www.asu.edu/admissions/contact/international.html (document id 939) -> **Cluster 1**
www.asu.edu/admissions/contact/index.html (document id 938) -> **Cluster 1**
www.asu.edu/admissions/contact/counselor.html (document id 935) -> **Cluster 1**

www.asu.edu/admissions/visitcampus/tempecampus/appointment.html (document id1075)-> **Cluster 2**

www.asu.edu/admissions/nondegree/screen6.html (document id 992) ->**Cluster 3**

For, total Docs=10 and k=5

www.asu.edu/admissions/international/faq.html (document id 963) -> **Cluster 0**

www.asu.edu/admissions/contact/transfer.html (document id 941) -> **Cluster 1**
www.asu.edu/admissions/contact/freshman.html (document id 937) -> **Cluster 1**
www.asu.edu/admissions/contact/familymember.html(document id 936)->**Cluster 1**
www.asu.edu/admissions/contact/international.html (document id 939) -> **Cluster 1**
www.asu.edu/admissions/contact/index.html (document id 938) -> **Cluster 1**
www.asu.edu/admissions/contact/counselor.html (document id 935) -> **Cluster 1**

www.asu.edu/admissions/visitcampus/tempecampus/appointment.html (document id1075)-> **Cluster 2**

www.asu.edu/admissions/nondegree/screen6.html (document id 992) ->**Cluster 3**

www.asu.edu/admissions/whychooseasu/index.html (document id 1081)->**Cluster 4**

Explanation: We can observe, that document id 1081 was in cluster 1 when value of k was 4, if we look at the URL pattern of document id 1081 (www.asu.edu/admissions/whychooseasu/index.html). Then, it appears that this document does not belong to cluster 1 since all the URL's in cluster 1 have common URL prefix i.e. www.asu.edu/admissions/contact/~ except doc id 1081. Hence, when we changed the value of k to 5 then document 1081 settled down in cluster 4, which looks more reasonable cluster for this document.

Similarly, When k=3 and total documents are 10 then, for query "medic care" document id 359(http://www.asu.edu/aad/manuals/acd/acd702-03.html) goes to cluster 1 which contains all URL's with prefix http://www.asu.edu/studentaffairs/mu/family/~, but when we change the cluster size to 4 (i.e. k=4) then this document goes to cluster 3 which is an appropriate cluster for this URL. This explains, as 'k' increases, we get tighter cluster.

Question) Do the clusters seem to roughly correspond to the natural category of the pages? Did the value of k affect this? Mention any other observations you have.
Answer:
Yes, Clusters correspond to the natural category of the pages. We will explain this by taking an example.

Example: For query “admissions”, we have taken k=5 and total docs=10.

www.asu.edu/admissions/international/faq.html (document id 963) -> **Cluster 0**
Title of webpage: **Undergraduate Admissions: International Admissions : FAQ**

www.asu.edu/admissions/contact/transfer.html (document id 941) -> **Cluster 1**
Title of webpage: **Undergraduate Admissions: Contact Admissions : Transfer**

www.asu.edu/admissions/contact/freshman.html (document id 937) -> **Cluster 1**
Title of webpage: **Undergraduate Admissions : Contact Admissions : Freshman**

www.asu.edu/admissions/contact/familymember.html(document id 936)->**Cluster 1**
Title of webpage: **Undergraduate Admissions : Contact Admissions : Family Member**

www.asu.edu/admissions/contact/international.html (document id 939) -> **Cluster 1**
Title of webpage: **Undergraduate Admissions: Contact Admissions: International**

www.asu.edu/admissions/contact/index.html (document id 938) -> **Cluster 1**
Title of webpage: **Undergraduate Admissions : Contact Admissions**

www.asu.edu/admissions/contact/counselor.html (document id 935) -> **Cluster 1**
Title of webpage: **Undergraduate Admissions : Contact Admissions : Counselor**

www.asu.edu/admissions/visitcampus/tempecampus/appointment.html (document id1075)-> **Cluster 2**
Title of webpage: **Undergraduate Admissions: Visit ASU: Meet an Admission Counselor**

www.asu.edu/admissions/nondegree/screen6.html (document id 992) ->**cluster 3**
Title of webpage: **Undergraduate Admissions: Non-Degree Seeking Students:**

www.asu.edu/admissions/whychooseasu/index.html (document id 1081)->**Cluster 4**
Title of webpage: **Undergraduate Admissions: Why Choose ASU**

Explanation:
If we look at the titles of webpages then we will observe that documents in cluster 1 are concerned with “contacting admissions” for various categories such as contacting Counselor, contact for internationals. Where as, document in cluster 4 is concerned with “why choose ASU” and not about contacting admissions.
Also, we checked all the web pages and K means clustering is picking the appropriate document for respective clusters.
Also, As the value of k increases we will get more logical/appropriate clustering but there are trade offs when value of k increases. For example, Time taken to iterate to find the appropriate grouping of documents for respective clusters increase as value of k increases. Also, more memory will required when value of k increases as we need to maintain more mean-Vectors/Centroid-Vectors for ‘k’ clusters. We can conclude, as value of ‘k’ increase, we get more appropriate clusters but that comes with more computation time and memory.

Analysis of Snippet Generation Algorithm

1) Snippet generation: Snippet generation function extracts the snippets from the selected top webpages for a given query. We will discuss the purpose of snippets, algorithms to generate the snippets from the webpage and time taken to find snippets.

1.1) Purpose of generating snippets: Snippets of a webpage gives a useful insight to the user that why a particular page has been selected for a given query.

1.2) Algorithm and Data Structure to find the snippets: Algorithm to find the snippets from web page is defined below.

a) Get top 'k' webpages for a given query using TF-IDF similarity and store in an arraylist. Also, store document id and their corresponding URL's in a hashmap. We have taken top 10 web pages for our program.
b) Store the word frequency of query terms in a hashmap. Suppose, if a query contains two identical terms then term's frequency is set to 2.
c) Fetch URL from the HashMap and pass this URL to a function (read this URL from folder result3) which will read it, line by line. This function will convert the source of HTML into plain text using jSoup library and store it into a string builder object and pass it to another function.
d) Now, we will check which line contains the more occurrences of query terms and increment the score of line if it contains query terms. We will store lines and their respective scores in a hashmap.
e) Pick the top 2 lines from the hashmap based on their scores.
f) Append the second line if first line's length is less than 40.
g) Repeat Steps from c-f for 10 documents and store in a hashMap.
h) Display the URL's and their respective snippets.

Extra Feature: We are also picking the title of a webpage using a regular expression to display it with URL and its snippets just like how google shows its results.

1.3) Time Taken: Total time taken to create snippets is 4280 MS for query "admissions" where 4080 MS is taken by TF-IDF search and only 200 MS is taken to generate snippets and to find the title of those web pages.

For query "carl hayden":
Time taken for TFIDF search: 4046 MS
Time taken to generate snippets: 168 MS
Total Time: 4214 MS

For query "president":
Time taken for TFIDF search: 4055 MS
Time taken to generate snippets: 118
Total time: 4173 MS
We can conclude that snippet generation does not take too long if top k TFIDF documents are already given.

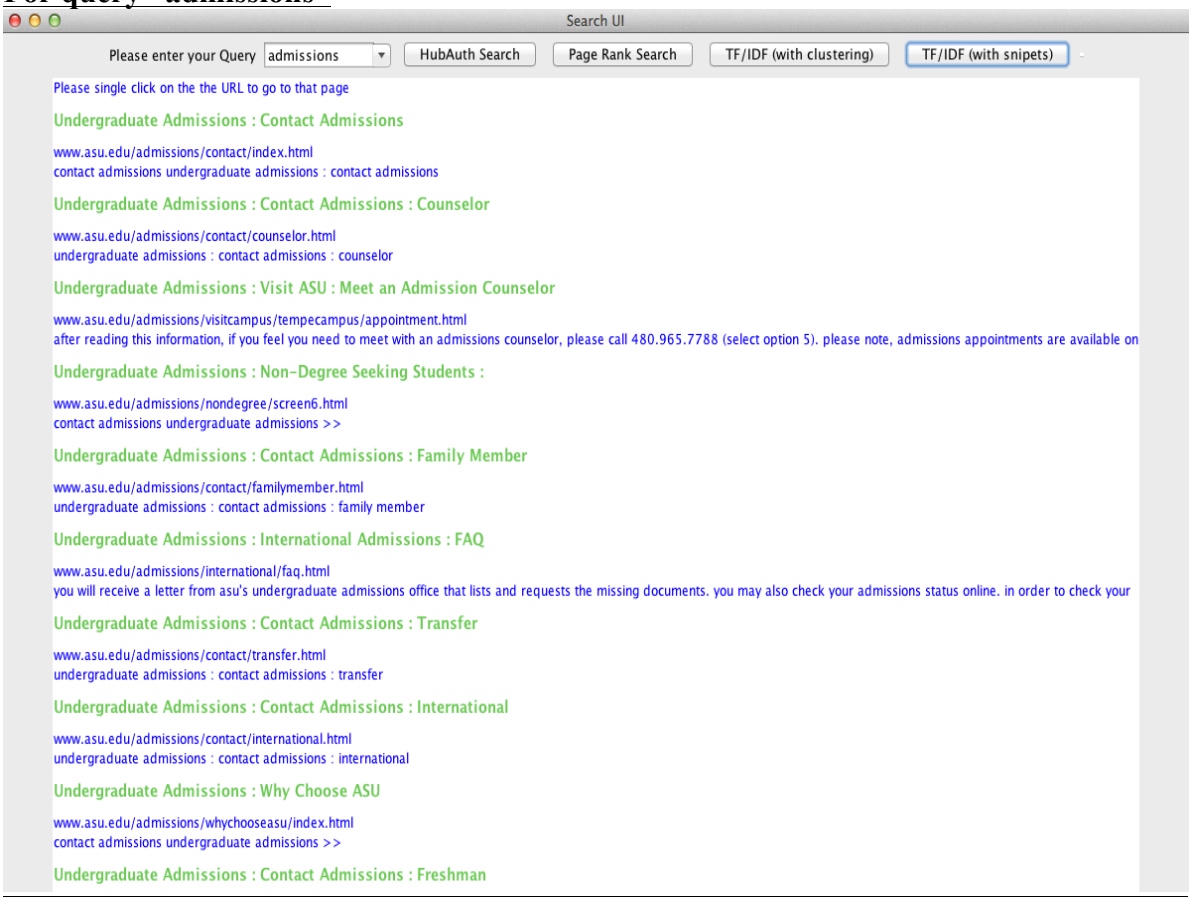
1.4) Relevance of Snippets: Since we have checked which lines in the document have the most occurrences of query terms then the snippet which is shown to the user will be

relevant to the user. The snippet algorithm is taking TF into consideration while generating snippets; snippets are fetched from the web page containing query terms hence user will find them relevant as algorithm is fetching all those lines which contains the most occurrences of query term.

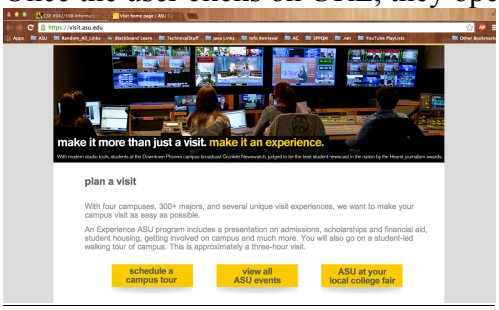
We are fetching titles of webpages as well; we could use title text to decide which title is more relevant to the query and present those webpages first whose title is more relevant to query terms. Please check snapshots for more clarity on the same. Further more, if can take IDF into consideration then terms which have low IDF values should not be shown to the user in the snippet which makes snippet more relevant.

Screen Shots of Snippet Generation:

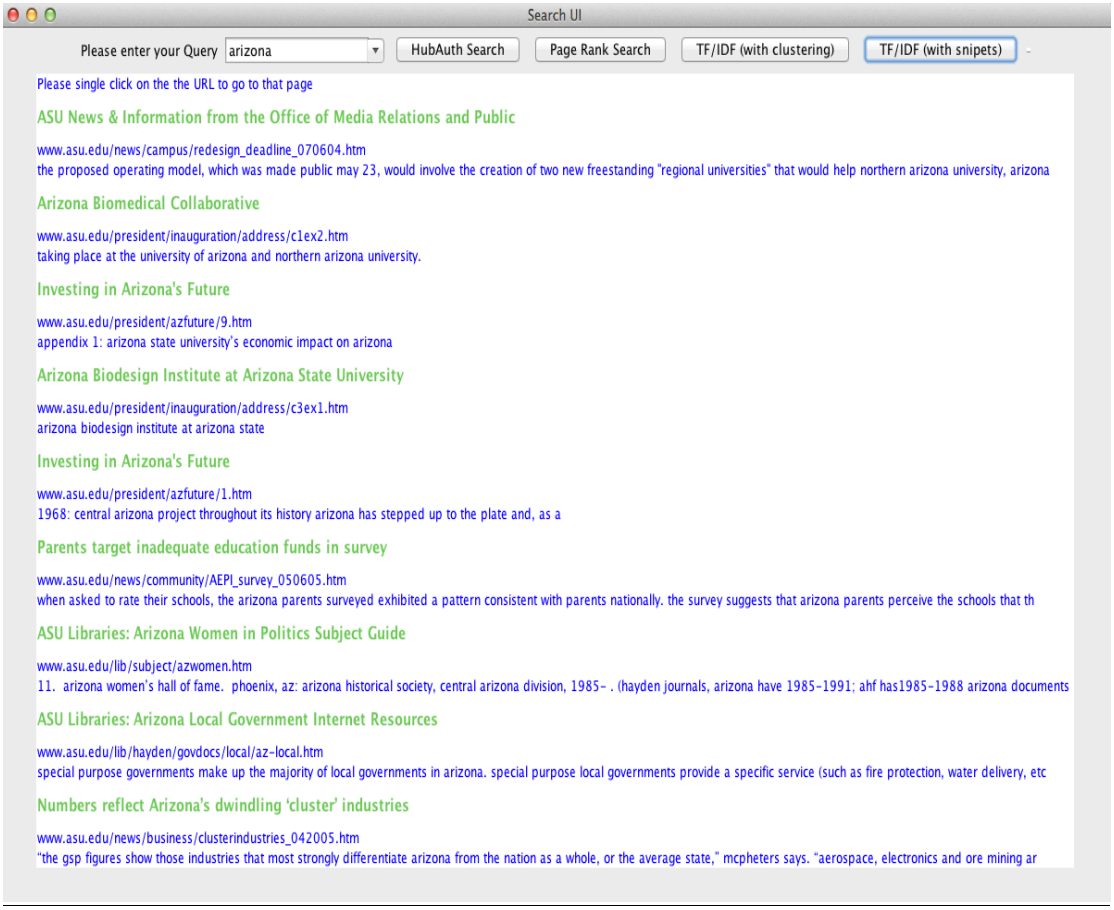
For query "admissions"



Above output shows that we are generating Title of a webpage, it's URL and its snippet. User gets a useful insight about a webpage.
Once the user clicks on URL, they opens up in a browser(shown below.)



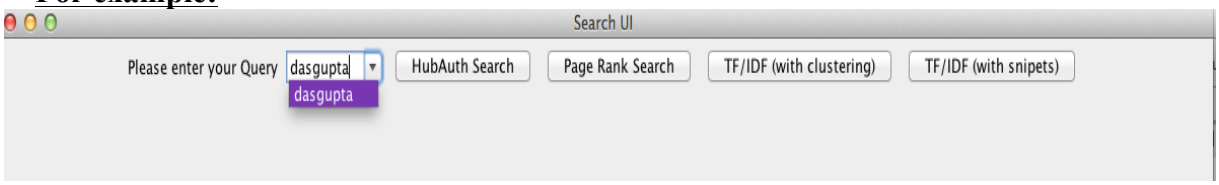
For query “arizona”:



Analysis of Scalar Clustering Algorithm

Scalar Clustering: We have implemented scalar clustering and used it to give user suggestion of closest word to what he types in search text box.

For example:



User enters “dasgupta” as query, and scalar clustering added “partha” as soon as user presses space button.

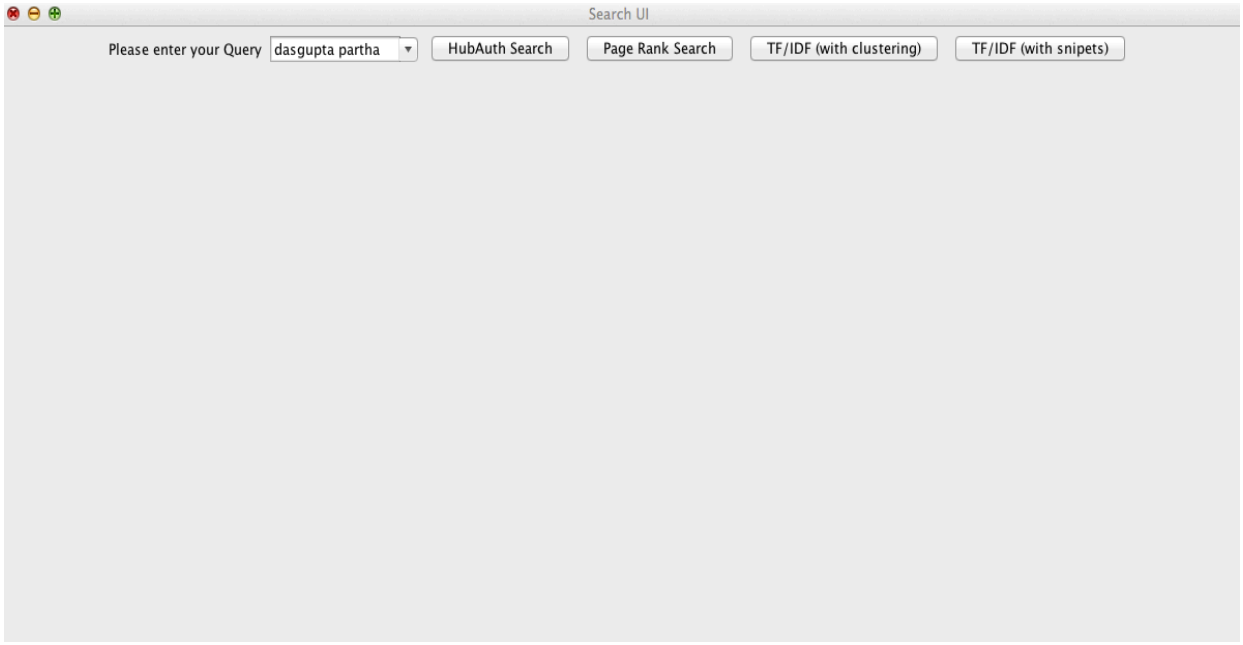
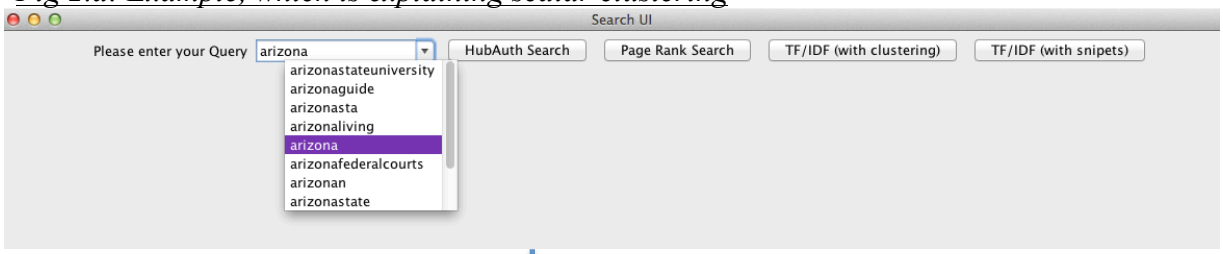


Fig 2.a: Example, which is explaining scalar clustering



Scalar clustering gives “state” as closer word to “arizona”

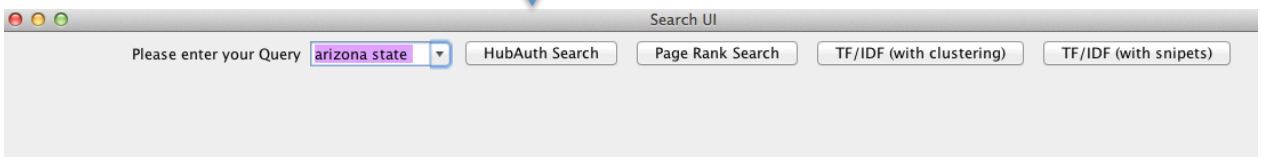


Fig 2.b: scalar clustering helping user to find out the most closer term

User Interface

We have implemented query completion and query checking as well, we will discuss more about them.

a) **Query Completion:** We have constructed a dictionary using a hashmap that contains all the terms, which are present in corpus. Every time, User starts typing a query word, key event gets fired and program will look for all the words which starts with the words entered by the user as shown below.

Screen Shots:

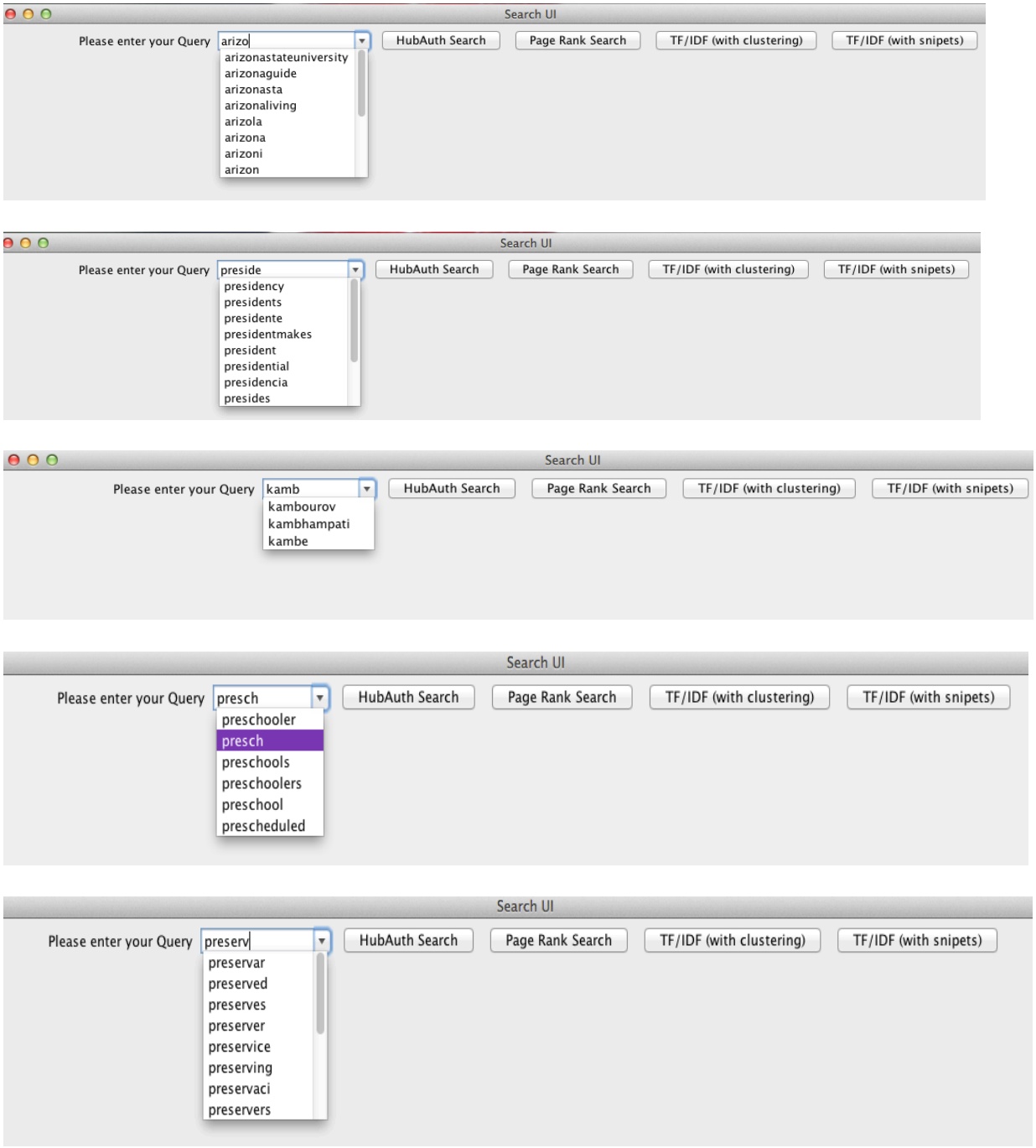


Figure: Query Completion

b) **Query Checking:** We are also checking if user has entered a wrong word. If user enters a query word then we will check if that query word is present in the dictionary(HashMap that contains all the corpus terms), if dictionary does not have that word then we will compute the levenshtein distance of that query word with all the terms in corpus. Whichever terms has the least levenshtein distance with the query word then we will suggest that term to the user like shown below.

ScreenShots:

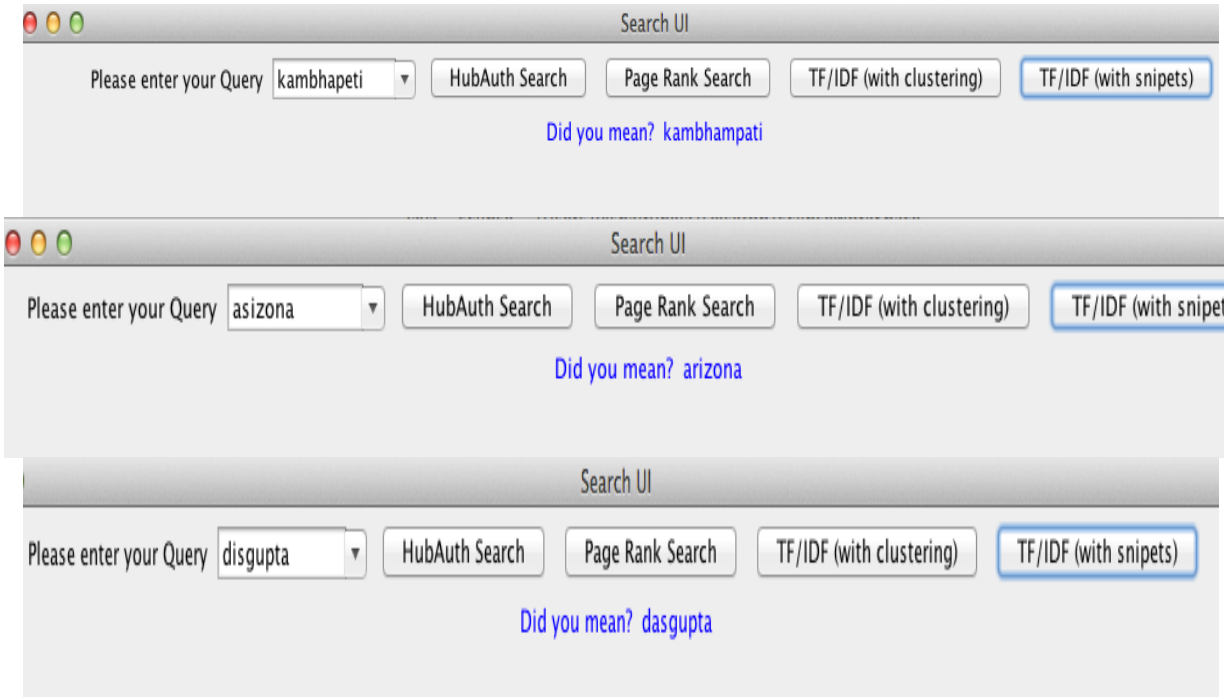


Figure: Query Checking

Algorithm for Query checking:

- Pre-generate a dictionary which contains all the terms present in corpus
- Get the query entered by the user
- Check if the query is present in the dictionary(created at step 1), if yes, find the TF_IDF similar documents, if no, then go to step 4
- Find the levenshtein distance of query word with all the word in dictionary.
- Which ever terms has the least levenshtein distance will be the term, user possibly intends to ask about
- Suggest the term to user

c) **Showing Title of web page:** We also generating titles of webpage so user can get an useful insight into what title of webpage is about as shown below.

Screen Shots:

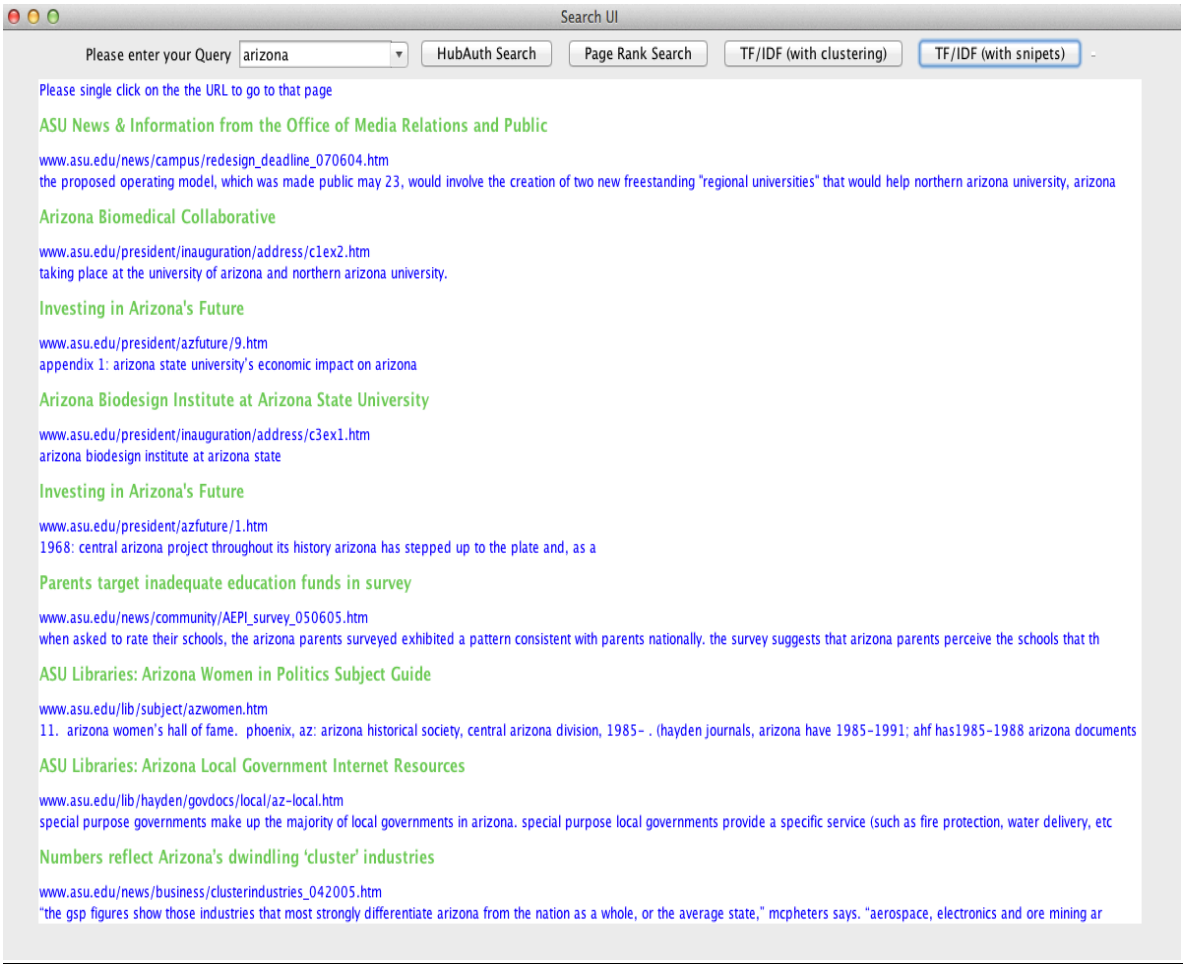
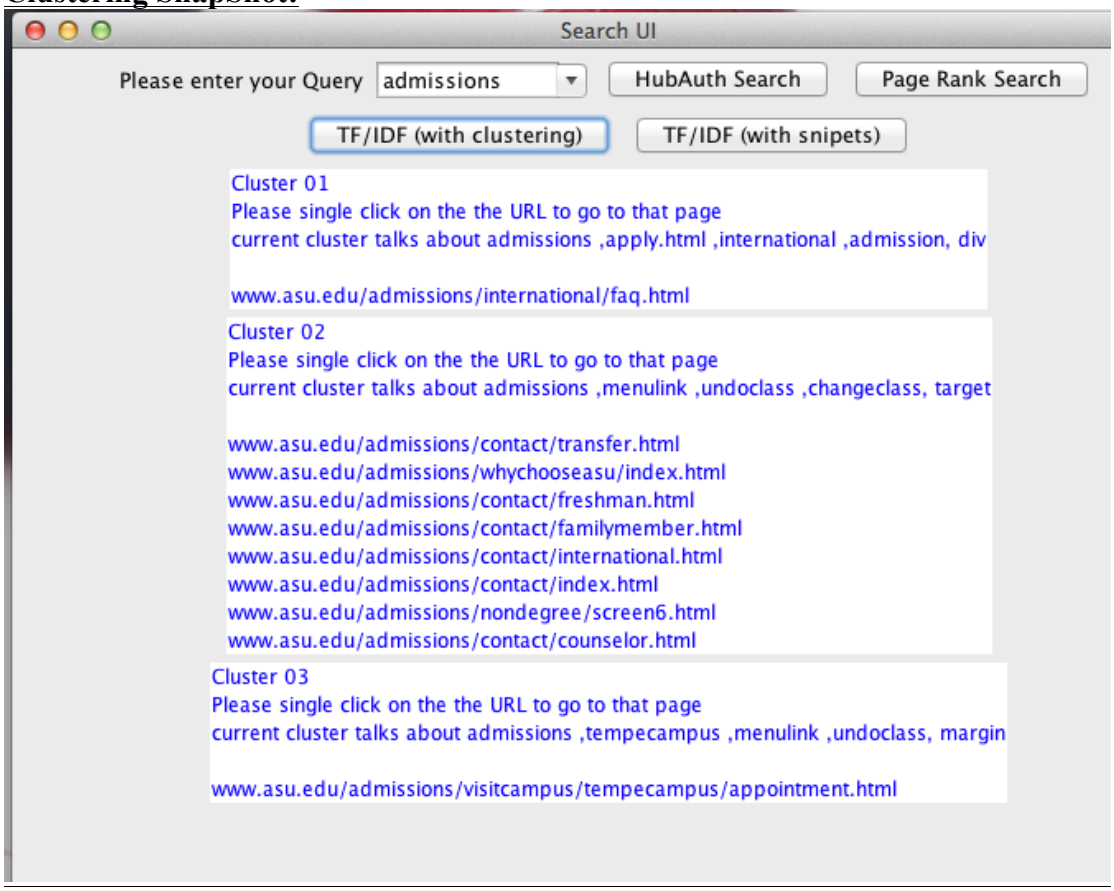


Figure: Title generation of webpages

Algorithm:

- Read a source page of an URL line by line
- Check if that line contains the title tags by pattern-matcher using a regular expression
- Update the document id and its corresponding title

Clustering SnapShot:



User can click on any link to go to that page.