**Project Part 3 Report**

**CSE 598: Information Retrieval**

**By:**

**Udai Arora**

**1206324102**

**Task 1: K-means**

* **5 marks:** Cluster the results of the queries given below, with
  + Number of documents clustered, N = 50
  + Number of clusters, k = 3
  + Similarity algorithm = Vector similarity (TF-IDF) **without** PageRank

and submit a printout of the document numbers of the top-3 documents in each cluster.

**Ans.** Using N=50, k=3, TF-IDF Similarity without link analysis and randomly picking the initial seeds, we get the following documents in the 3 clusters each:

|  |  |  |  |
| --- | --- | --- | --- |
| Query | Cluster 1 | Cluster 2 | Cluster 3 |
| Medic care | 23333  23365  23367 | 23565  359  23559 | 20952  22816  19875 |
| Employee benefits | 4591  787  24959 | 4543  775  4642 | 4599  223  4592 |
| Parking decal | 2360  2366  2370 | 649  2406  4595 | 2282  2287  20828 |
| Admissions | 1081  1087  1088 | 938  935  992 | 1075  1043  1048 |
| languages | 14437  14439  14441 | 1374  14358  14421 | 20928  20888  70 |

* **3 marks:** For each cluster you obtained above, determine short "summaries" of the clusters, using keywords that most distinguish those clusters from other clusters. Explain how you obtained these summaries. You are free to come up with your own strategy for finding these summaries, there is no set algorithm that you *have* to use.

Ans. The cluster Summaries corresponding to the above queries/clusters are shown below:

|  |  |  |  |
| --- | --- | --- | --- |
| Query | Cluster 1 | Cluster 2 | Cluster 3 |
| Medic care | Health; pharmacy; student affairs | Child; leave; family | news; nursing; nurses |
| Employee benefits | Leave; care; absence | Retirement; certificates; recognition | Layoff; rules; regular |
| Parking decal | Transit; disabled; rates | Vehicle; gate; services | Improvement; maintenance; news |
| Admissions | Why choose asu; undo class; change class | international; how to apply, transfer | Steps; visit campus; orientation |
| languages | Slavic; language; Russian | Udm; literature; placement | Certificates; studies; minor |

How these summaries were obtained: After the clustering was complete, for each of the clusters, the final centroid vector was taken. These vectors were now placed in an array and sorted according to the tf\*idf values of each of the terms. The sorted words were used to generate each clusters summary. The words were also filtered based on a stop word list (which contained most of the html tags and elements) and also making sure that the word chosen for a cluster doesn’t appear in any of the clusters before it. This is how unique keywords for each clusters were generated. The algorithm used in the program has a time complexity is O(k\*t\*log(k)) assuming a constant number of keywords are generated. Here k(for number of clusters) is the outer loop, t is the vocab of that cluster and log(k) is to filter out duplicates so as to make sure every cluster has a unique summary.

* **8 marks:** Pick any two queries from the set given below. Change the value of 'k' between 3 and 10. What do you observe? Why?
  + How does execution time change?
  + How does the similarity of the document to the centroid of the cluster change?
  + How did the value of k affect the clustering? Justify with a couple of examples.
  + 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.

**Ans**

1. The execution time is shown in **milliseconds** in the following data:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| k= | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| admissions | 574.078 | 631.4946 | 664.3534 | 706.6205 | 777.184 | 851.4254 | 957.3054 |
| languages | 610.976 | 700.399 | 730.5508 | 739.6067 | 802.8238 | 905.1652 | 1034.427 |

A graph for the same is shown below:

Series 1 correspond to languages and series 2 to admissions.

Hence as we see, the execution time has increased almost linearly. This is because the algorithm used has a time complexity of n\*t\*k. Since, n and t are essentially constants and assuming approximately the same number of iterations, time taken is directly proportional to the value of k. Hence time taken increases with increase in k.

1. We know from Rao’s Clustering Slide #18 that similarity between a document and the centroid is equal to the average similarity between that document and every other document. We also know that average similarity between all pairs of documents in a cluster corresponds to the square of the magnitude of the centroid vector for that cluster. Hence, looking at the square of centroid vectors’ magnitude for different values of k:

Query: ”admissions”

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| k= | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| |centroid|^2 | 42444.42 | 44212.24 | 46511.64 | 49198.35 | 51895.43 | 55323.55 | 57852.19 |

Query: ”languages”

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| k= | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| |centroid|^2 | 33289.33 | 74100.67 | 114933.9 | 142298.4 | 178077.5 | 200162.2 | 246461.3 |

Hence as we see in the graphs above**, with the increase in the value of k, the average similarity of the documents with the respective centroids also increases**. This is because the clusters get tighter with increase in k. When k=1 (that is no clusters), the similarity of the documents with the centroid will be the lowest. As k increases, this similarity also increases. Hence when k=n (no of documents), each document will be in its own cluster and will have maximum similarity to the centroid. So the graph above are expected.

1. In general, it was observed that higher the value of k, tighter the clusters.

Consider this example where the query used is “parking”

**K=2**

Cluster 1 (pages about news on parking)

<http://www.asu.edu/news/campus/parking_construction_082403.htm>

<http://www.asu.edu/news/stories/200604/20060407_parking.htm>

<http://www.asu.edu/news/forthemedia/visitingcampus.htm>

Cluster 2 (pages about visitor/event parking)

<http://www.asu.edu/dps/pts/event/gammage.html>

<http://www.asu.edu/dps/pts/parkingreg.htm>

<http://www.asu.edu/dps/pts/visitor/media.html>

**k=5**

Cluster 1(pages about news on parking)

<http://www.asu.edu/news/campus/parking_construction_082403.htm>

<http://www.asu.edu/news/stories/200604/20060407_parking.htm>

<http://www.asu.edu/news/forthemedia/visitingcampus.htm>

Cluster 2 (Pages about transit and prices)

<http://www.asu.edu/dps/pts/taskforce_home.htm>

<http://www.asu.edu/dps/pts/questions_answers.htm>

<http://www.asu.edu/hr/training/learning_center.html>

Cluster 3 (Pages about parking during new admissions and orientation)

<http://www.asu.edu/graduation/geninfo/parking.html>

<http://www.asu.edu/ssc/commence/geninfo/parking.html>

<http://www.asu.edu/admissions/orientation/tempe_parking.htm>

Cluster 4 (Pages about event and visitor parking)

<http://www.asu.edu/dps/pts/event/gammage.html>

<http://www.asu.edu/dps/pts/visitor/media.html>

<http://www.asu.edu/dps/pts/event/index.html>

Cluster 5 (Pages about parking services, maintenance and decals)

<http://www.asu.edu/dps/pts/parkingreg.htm>

<http://www.asu.edu/aad/manuals/pts/index.html>

<http://www.asu.edu/dps/pts/decals/crossparking.htm>

Hence as we see in the example above, as we increase the size of clusters, the clusters becomes more specific to the documents. For example, the documents in cluster 3 for k=5 are about parking during admissions and orientation. These documents were a part of cluster 2 for k=2 and were classified previously under visitor/event parking. Increasing k has helped classify these documents more accurately.

1. As seen above, the clusters do correspond to the category of pages in them. Increasing the value of k makes clusters tighter resulting in more accurate representation of category of pages in them.

For example, the query “contact” was run for 2 values of k

For k=3

Cluster 1 (student affairs; internships; employement)

<http://www.asu.edu/studentaffairs/mu/community/current_partner.htm>

<http://www.asu.edu/studentaffairs/mu/community/monthly_service.htm>

<http://www.asu.edu/studentaffairs/mu/community/internships.htm>

Cluster 2 (Admissions; Loans)

<http://www.asu.edu/admissions/contact/index.html>

<http://www.asu.edu/fa/email/PerkinsNoMPN.html>

<http://www.asu.edu/clas/communication/contact/index.html>

Cluster 3 (classroom; admissions)

<http://www.asu.edu/admissions/ASUNearYou/asucontact.html>

<http://www.asu.edu/president/contact/index.html>

<http://www.asu.edu/studentaffairs/ed/iso/old/student_org.htm>

For k=5

Cluster 1 (uaoi office; student affairs)

<http://www.asu.edu/studentaffairs/mu/community/current_partner.htm>

<http://www.asu.edu/uoia/contact_txt.html>

<http://www.asu.edu/provost/spiada/uoia/contact_txt.html>

Cluster 2 (loan; social sectrity)

<http://www.asu.edu/fa/email/PerkinsNoMPN.html>

<http://www.asu.edu/fa/email/PerkinsNoMPNorInt.html>

<http://www.asu.edu/fa/email/AcceptPerkins.html>

Cluster 3 (admissions; classes)

<http://www.asu.edu/admissions/contact/index.html>

<http://www.asu.edu/admissions/ASUNearYou/asucontact.html>

<http://www.asu.edu/studentaffairs/ed/iso/old/student_org.htm>

Cluster 4 (hr; people)

<http://www.asu.edu/president/contact/index.html>

<http://www.asu.edu/clas/communication/contact/index.html>

<http://www.asu.edu/clas/communication/contact/index.html>

Cluster 5 (on campus jobs; internships)

<http://www.asu.edu/studentaffairs/mu/community/monthly_service.htm>

<http://www.asu.edu/studentaffairs/mu/community/internships.htm>

<http://www.asu.edu/studentaffairs/mu/community/service.htm>

Hence, we can make out from the url’s that the documents correspond to the clusters they are in. Increasing k causes the clusters to be more specific and the cluster summary corresponds better to the gist of the documents in them.

Another Observations: The subdomains seem to be the same for clusters as k gets higher. This is because generally, subdomains correspond to particular topics. Increasing k causes them to be grouped into the same cluster.

* **4 marks:** Submit your code with comments.

**Ans.** The code with comments has been emailed.

**Task 2: Final Integrated Search Engine Interface**

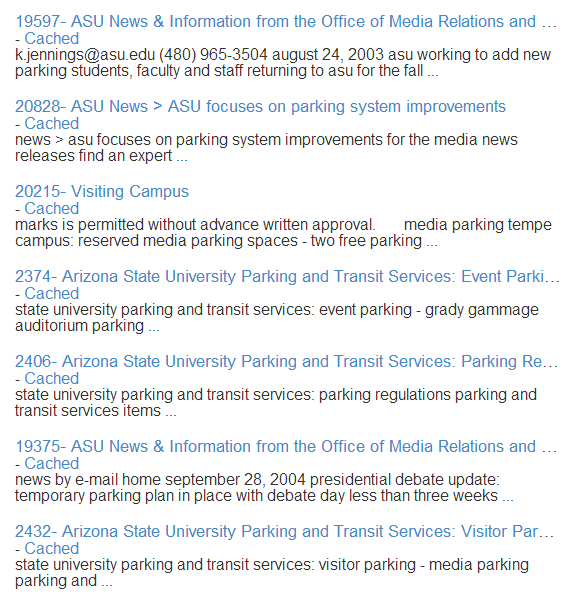
**20 marks:** Points will be awarded for implementation, code, performance and a thorough analysis.

1. **Snippet generation:**for every result document retrieved by TF/IDF, generate a snippet of text. The snippet must be a block of text from the document that helps the reader understand why the document is relevant to the query. Analyze the time taken to generate the snippet, describe the data structures and algorithm you used and judge the relevance of the snippet.
2. Implement **scalar cluster analysis** to suggest ways of elaborating the query terms.
3. **Improved GUI:** Here you will be evaluated on anything you do over and above the average GUI submitted for Phase 2 project. Improvements may involve (1) just showing the snippets and query elaboration and/or (2) also showing the clusters in the results and/or (3) adding more fancy things as AJAX based query completion etc.

Ans.

1. Snippet Generation: On an average, the time taken to generate 100 snippets is 89.45332 milli-seconds, 200 snippets is 163.68699 milli-seconds and 400 snippets is 344.12245 milli-seconds. This means a snippet is generated in less than a millisecond.
2. The algorithm used gives more weights to query terms having higher idf when choosing snippets. As soon as the term with highest idf is found, term in the proximity of it are also added to the snippets. If the highest idf term is not found, next highest word is searched and the process continues.

Top Snippets for “parking”:



The complexity of the algorithm is n\*w\* q(n= no of documents for which the snippet is generated, w= words in the document, q= query terms). This is because there is an outer for loop for number of documents which calls the function to generate snippets. The function has an outer for loop over the words in the document and an inner for loop over the query terms. The idf values are obtained from a map and terms are stored in a priority queue. The words of the documents are split using a regex (“ +”) and stored in an array. The snippet words are also stored in an array. Jsoup library is used to read documents.

The generated snippets are highly relevant to the query as shown for the query above. The first document talks about the person who is working to add more parking and gives their details in the snippet itself. The second document is about improvements on paring system and the snippet seems to agree with that. The third one talks about visiting the campus and the snippet gives a gist of visiting the campus and reserving parking when you do so. This is true for all the other snippets as well.

1. Scalar Cluster analysis was implemented. This was done using creating the Term-Term matrix using the top 10 similar documents of the results. After that, the dot product is taken and normalization is done following the steps in Rao’s slides on scalar clustering. An example for scalar clustering is shown below. This is the suggestion for the query “parking”. Clicking on any of these term were cause them to be added to the query and the query to be run again.



The time complexity of the algorithm is d\*t where d=no. of top documents used to construct index and t= vocab of those documents.

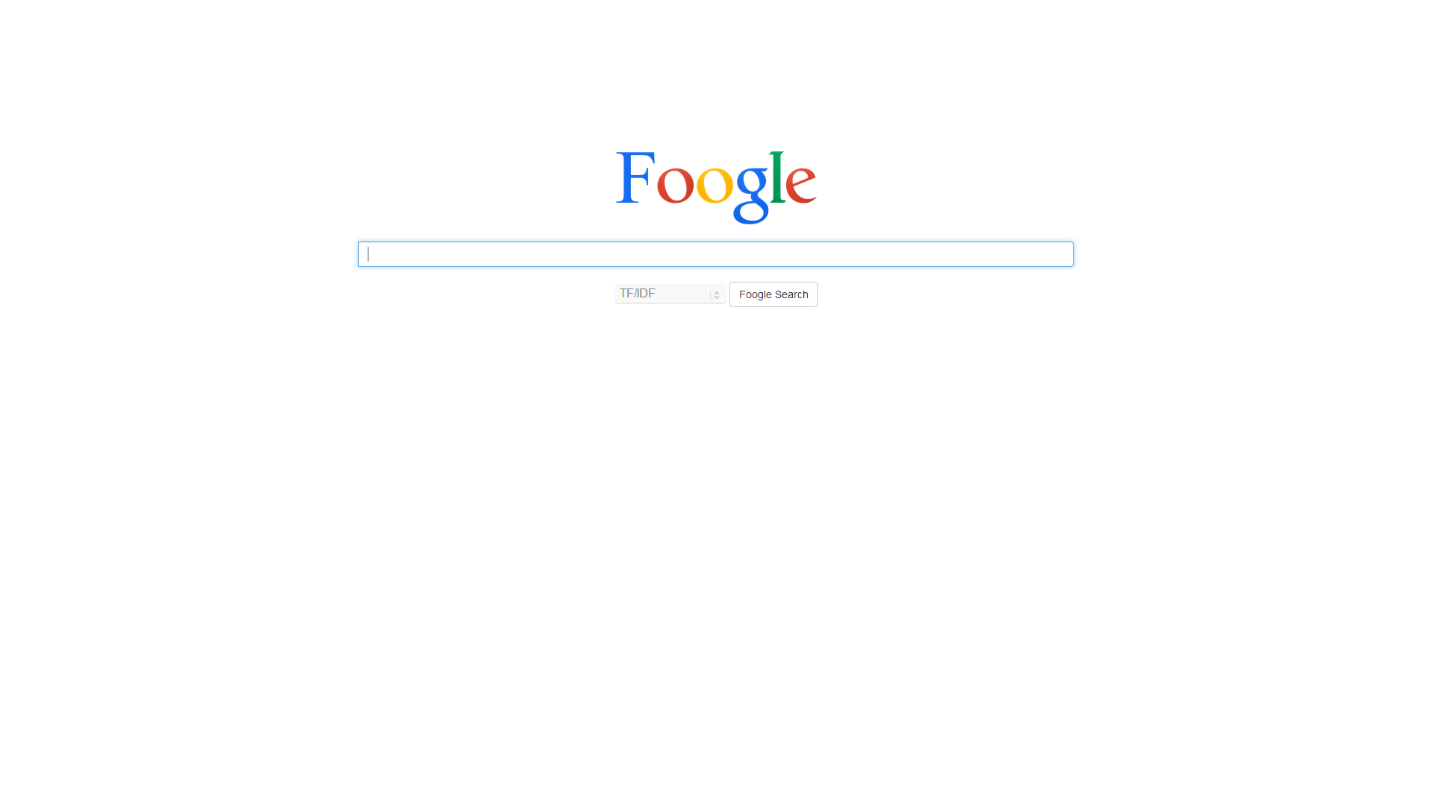
The average time taken to generate scalar clusters for query elaboration is 5241.231341 milliseconds (for n=10)

1. The GUI has been drastically improved. GUI is built using HTML5, CSS3, Javascript, Jquery, Twitter Bootstrap and a few other JavaScript libraries. A RESTful service based backend was implemented using Jersey and the Web Page calls this backend using AJAX calls.

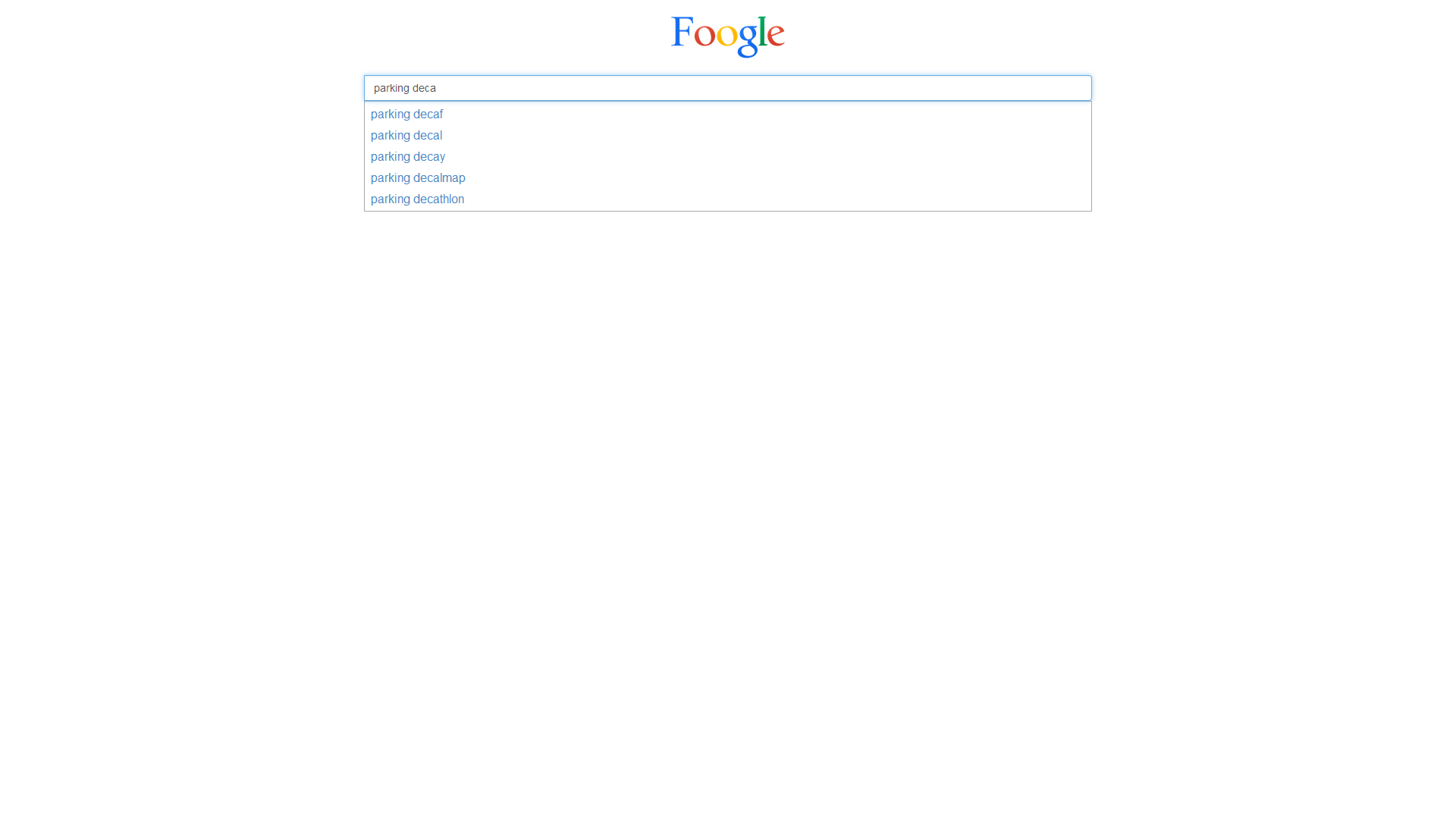
The following screen shows the page which opens on the first run (Pre-computations)



After the loading completes (usually takes 5-6 seconds), the search screen displays as shown below:

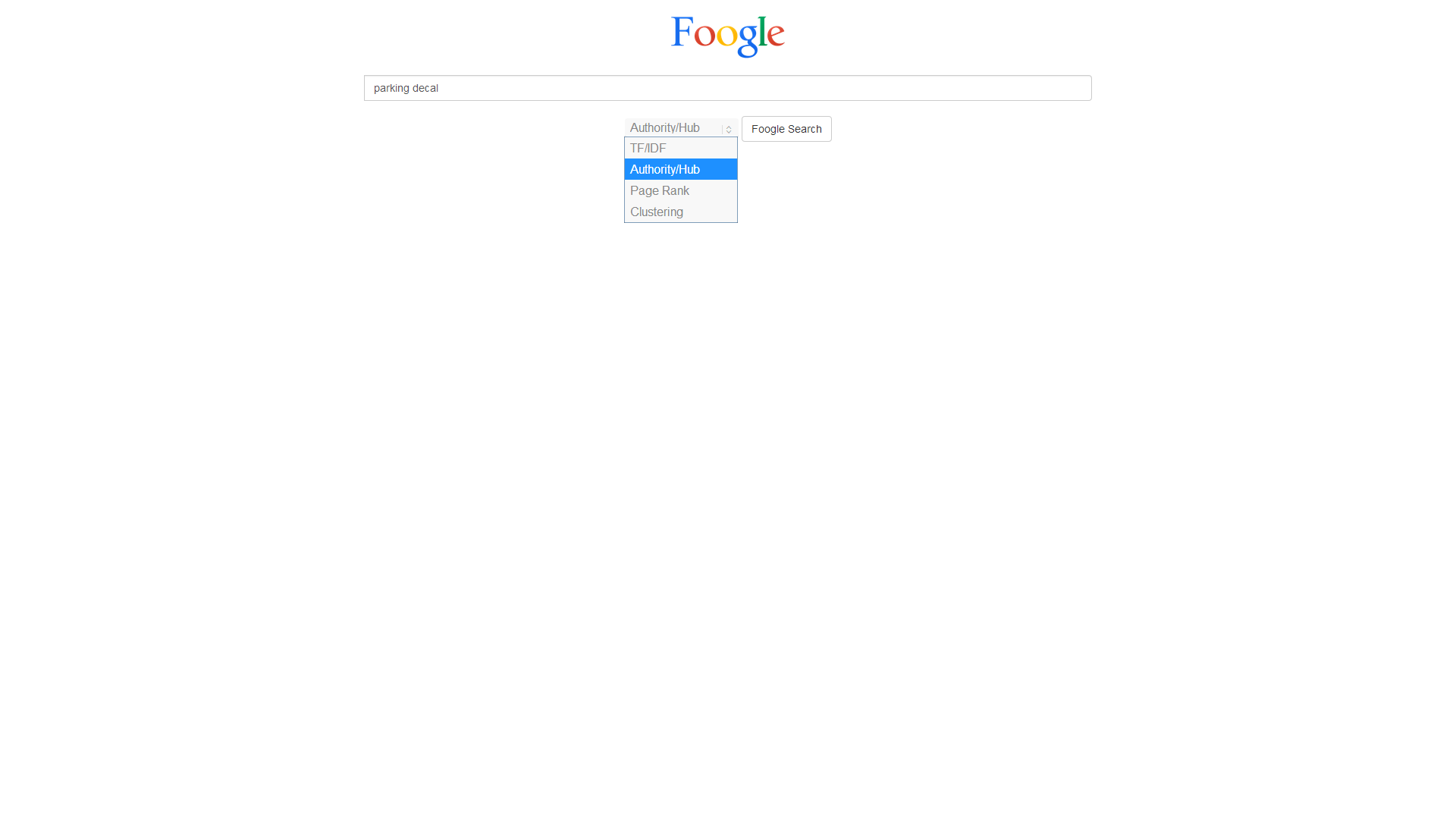


The search screen follows Google’s design idea of “Neat and Clean”. Some CSS3 transition effects have also been added. Autocomplete has been implemented using AJAX calls. The following screen shows autocomplete in action:

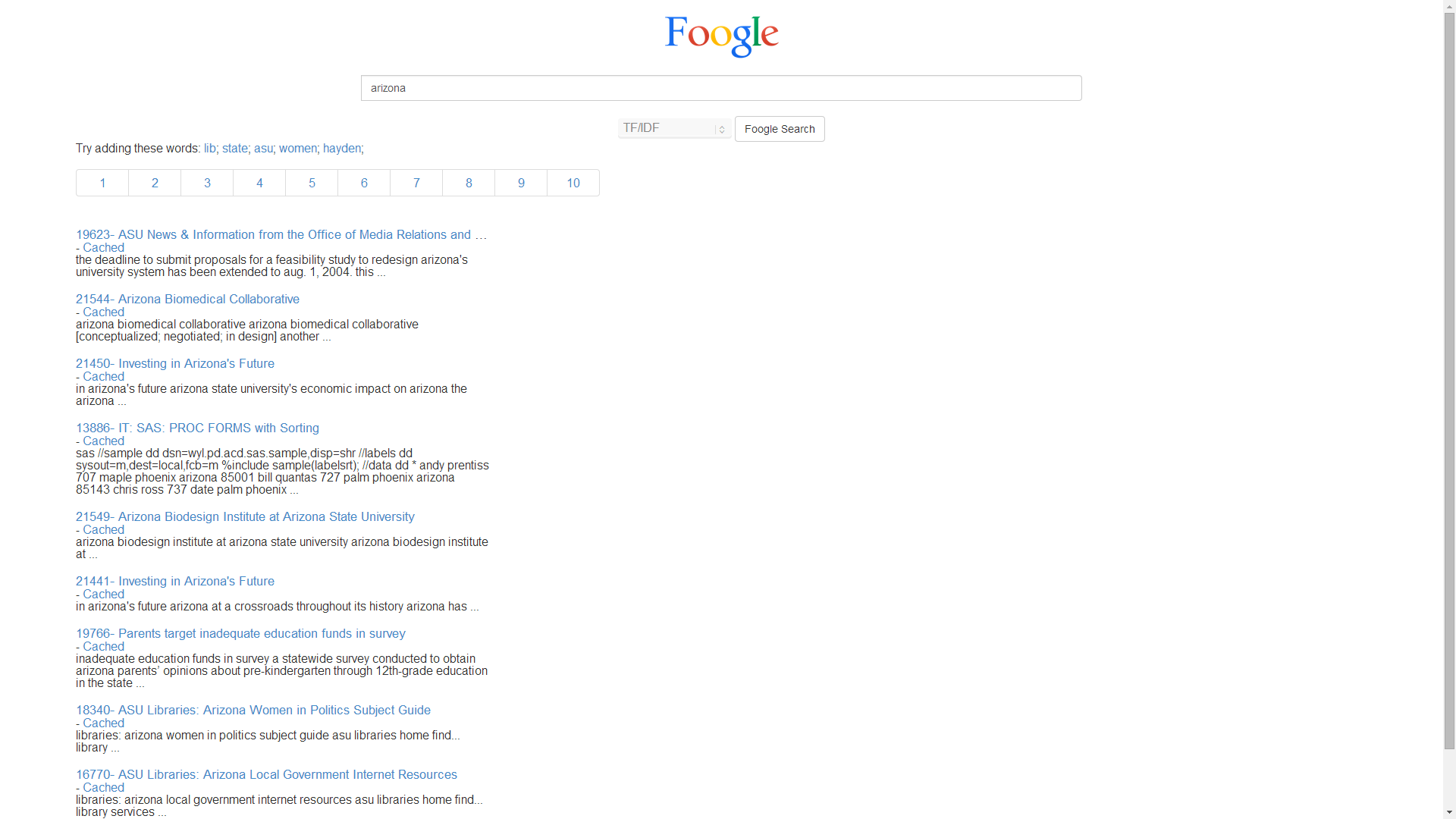


This autocomplete uses JQuery UI. One can use arrow keys to select the desired input. The autocomplete is implemented by sending an ajax request to the server with the value of the input field, extracting the last term (term being typed) from this on the server side and using regular expression to match it with the term in the idf map. The top 5 matched terms are returned. Since lookup in hashmap is of the order O(1), it **takes constant time** to get the results.

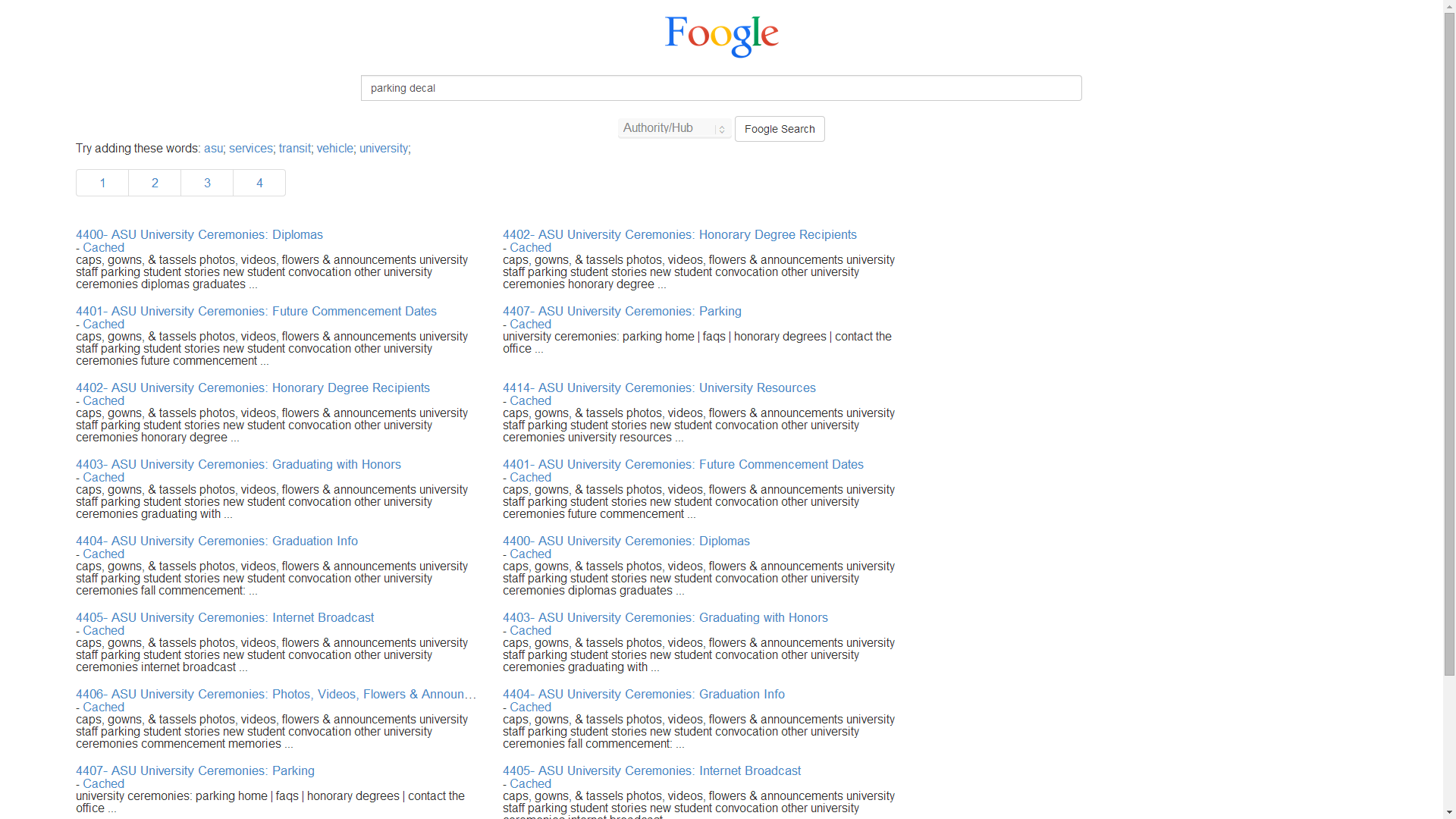
The screen below shows the available options which can be performed:



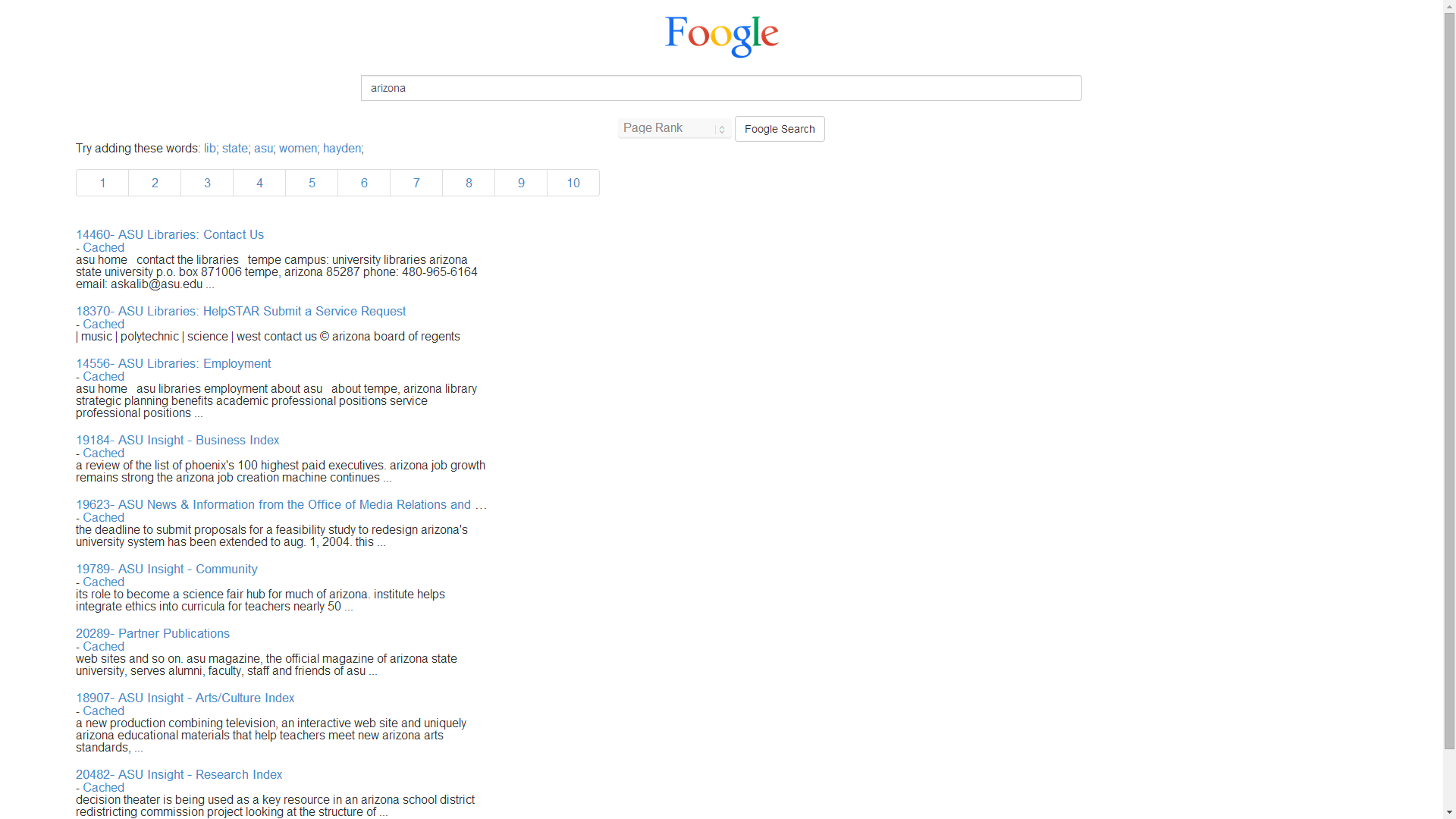
TF/IDF Results are shown below:



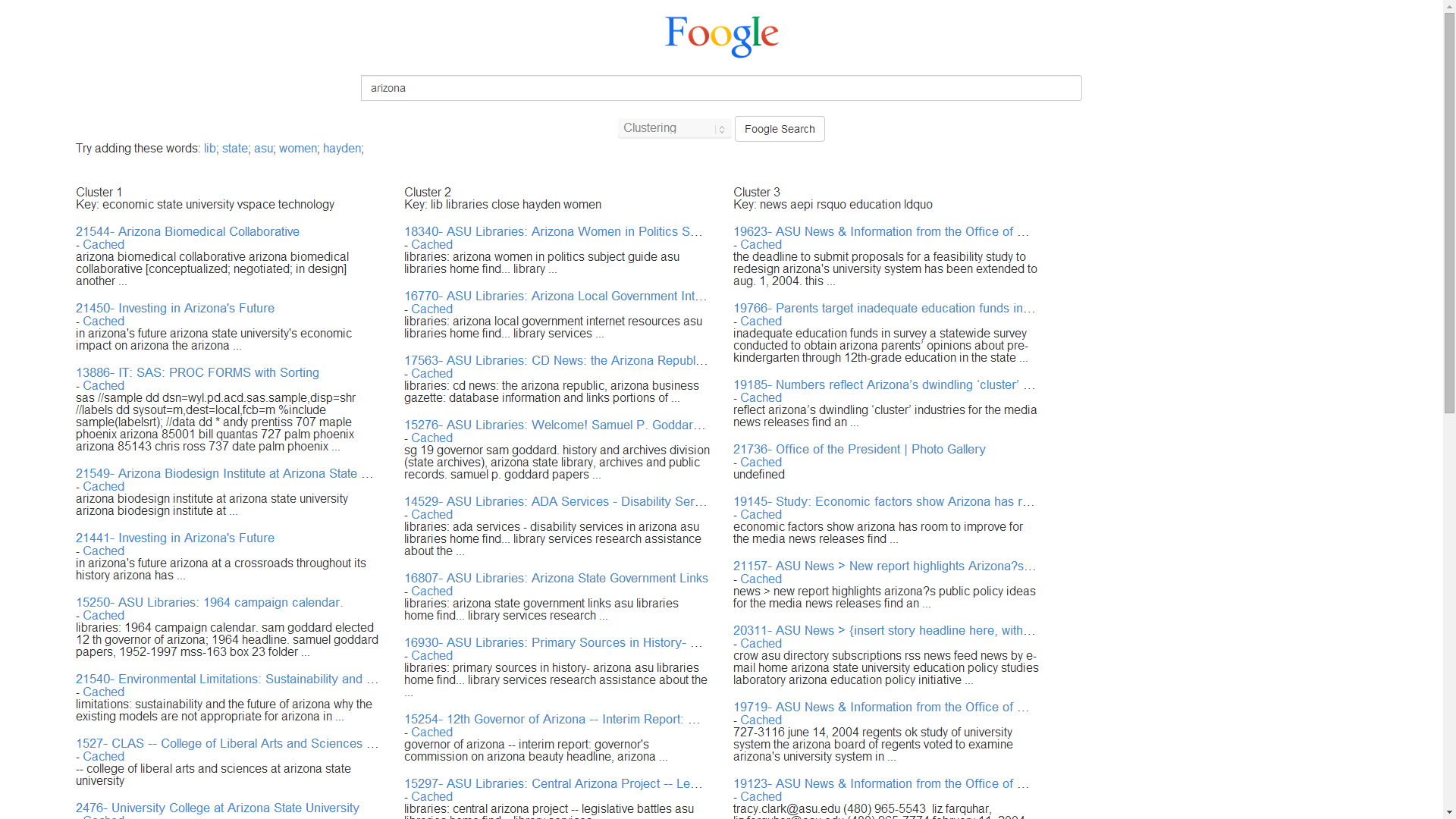
Results of Authorities and Hubs are shown on the left and right respectively:



Following are the page rank results:



Cluster using K-means was done and following are the results:



As it can been seen in the screenshots above, all the options (TF/IDF, A/H, Page Rank, Clustering) have scalar clustering suggestions, title and snippets for the documents and pagination if there are more than 10 results. All the links are clickable and redirect to the appropriate online URL. Since some of the pages have been removed since a crawl of all the pages was obtained, a cached copy can be seen by clicking on “Cached” label below the title.