Project Phase III

Nishant Bansal

		Medic Care	Employee Benefits	Parking decal	Admissions	Languages
Centroid 1	1.	23333	4599	649	938	14437
	2.	23365	4543	2406	1075	1374
	3.	21544	223	648	1077	14439
	Summary	Pharmacy, nurseclinic, medicalcare, patients, smoking	Certificates, retirement, recognition, classification, employee	Improvement, maintenance, decal, vehicle, disabled.	Visitcampus, orientation, admitted, visits, prospective	Dll, clas, menu, udm, small
Centroid 2	1.	23565	14460	639	935	20928
	2.	22816	789	2347	992	16277
	3.	23559	203	641	936	1869
	Summary	Child, workatasu, referral, provost, index	Workingatasu, hearing, progressive, supervisor, lib	Event, grady, gammage, transit, manual.	International, application, non, seeking, degree	Blockquote, buttons, labriola, url, hayden
Centroid 3	1.	20952	4591	4595	1081	20888
	2.	359	4592	2282	1087	5370
	3.	19875	4590	2287	1088	19929
	Summary	News, leave, nursing, employee, boolean	Leave, benefits, employees, salary, period	Margin, face, news, downtown, campus.	Whychooseasu, stories, quot, story, traditions	Certificate, eeeeee, minor, studies, lib
Related terms (Scalar clustering)		Subscriptions, more, new, research, background	Retirement, certificates, recognition, classification, system	Vehicle, transit, pts, strong, state	Undoclass, changeclass, titles, class, margin	Slavic, literatures, repeat, programs, new

Table. 1

Q: 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.
A: The top 3 results for the queries are mentioned in table 1.

Q: 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.
A: Please refer table 1 for the summaries. The algorithm used is described as follows:

- We have computed k centroids to do the clusters. The algorithm uses these centroids vector (c_j) which are in term space to distinguish the most dominant terms for the clusters.
- For each cluster j, we compute the effective vector for this cluster which is defined below

  Effective vector (a_j) = $c_j - \sum c_i$, where i = 0, 1, …., k and i ≠ j.
- From these effective vectors we then get the 10 most dominant terms (i.e terms with maximum coefficient in this vector). We then compute idf*coefficient for each term and return the top 5.

- Filtering is done to get better results. Initial filtering based on the value of IDF. Basically discard all the terms which are very frequent (with idf < 0.5) and terms which almost never occur (with idf > 6). Also discard some other terms like shall, will, he, she etc. to get better results.

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

A: For each computation of K-mean clustering, the program takes 10 random initial seeds and returns the best minima among them. As the seeds are randomly chosen for each set of random initial seeds the convergence is achieved in different number of iterations. So, it makes sense to consider the total count of iterations taken to give the output and also an average of many runs for a value of k rather than single value. The computation and results are shown below.

	K = 3	**K = 4**	**K = 5**	**K = 6**	**K = 7**	**K= 8**	**K = 9**	**K = 10**
Time taken (in milliseconds)	1258	1325	1830	1730	1995	1960	2270	2247
	1829	1939	1665	1975	2170	2152	2153	2065
	1389	1410	1881	1871	1987	2178	2050	2126
Average Time	1492	1558	1792	1858.67	2050.67	2096.67	2157.67	2146
Total count of iterations	34	41	42	37	42	38	48	44
	40	44	39	41	43	42	40	38
	38	36	43	40	45	43	39	43
Average count	37.33	40.33	41.33	39.33	43.33	41	42.33	41.67
Tightness of clusters* (summation of similarities)	24.20273	24.24652	24.43109	24.67128	24.78555	24.92295	24.88736	25.11541
	24.21950	24.36745	24.30759	24.42855	24.59085	24.69692	24.72416	24.81364
	24.14988	24.27266	24.42083	24.41365	24.87931	24.54492	25.05944	24.96243
Average tightness	24.1907	24.29554	24.3865	24.50449	24.7519	24.7216	24.89032	24.96383

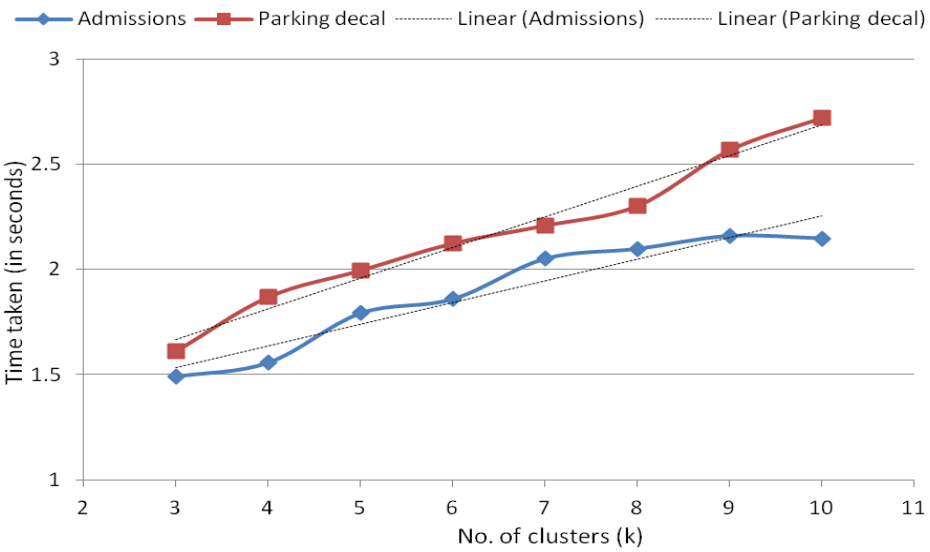
**a cluster is tighter, if the summation of similarity increases (because the documents in the cluster are more similar to the centroid.) This value is the summation of similarity between all the document to the cluster centroid. It is un-normalized value.*

Table. 2(a): Data for query "**admissions**"

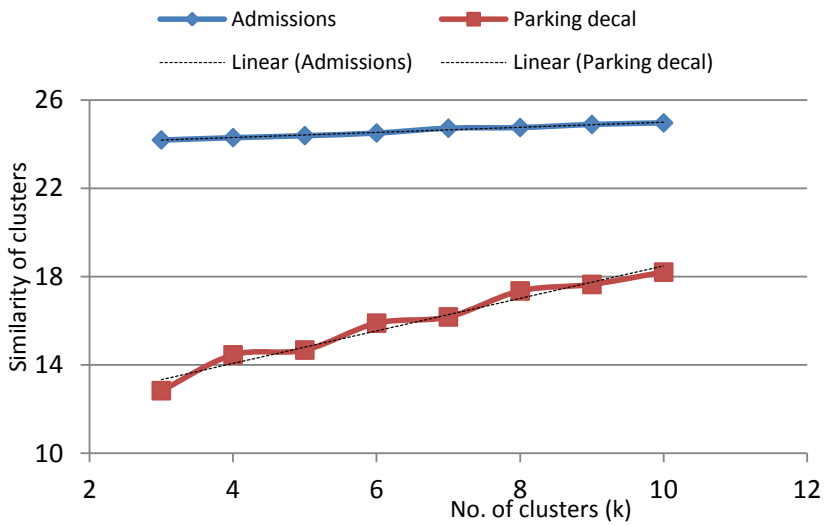
	K = 3	**K = 4**	**K = 5**	**K = 6**	**K = 7**	**K= 8**	**K = 9**	**K = 10**
Time taken (in milliseconds)	1484	1797	2100	2498	2335	2446	2681	2886
	1614	1858	1861	2018	2111	2198	2345	2501
	1740	1955	2026	1858	2180	2263	2675	2773
Average Time	1612.67	1870	1995.67	2124.67	2208.67	2302.33	2567	2720
Total count of iterations	42	47	51	57	51	50	54	53
	50	47	46	47	47	57	49	51
	51	52	50	42	54	48	54	54
Average count	47.67	48.67	49	48.67	50.67	51.67	52.33	52.67
Tightness of clusters* (summation of similarities)	12.84911	14.62440	15.15829	15.90214	16.71202	17.13680	17.51550	18.00235
	13.11840	14.37235	14.69805	15.83017	15.61894	17.07342	17.82247	18.11356
	12.53978	14.37472	14.17630	15.94354	16.19556	17.84658	17.60124	18.50172
Average tightness	12.83576	14.45716	14.67755	15.89195	16.17551	17.35227	17.6464	18.20588

**a cluster is tighter, if the summation of similarity increases (because the documents in the cluster are more similar to the centroid.) This value is the summation of similarity between all the document to the cluster centroid. It is un-normalized value.*

Table. 2(b): Data for query "**parking decal**"



Plot1: time vs no. of clusters



Plot2: similarity vs no. of clusters

(a) How does the execution time change?

A: The execution time increases linearly as can be observed from table:2(a), table:2(b) and plot1.

This happens because as the value of k (i.e the number of clusters) increases, each document has to be compared to all these possible centroids. The running time of the algorithm is O(kn), where k is the number of clusters and n is the number of documents to be clustered. So, for constant number of documents the running time increases with increase in 'k'.

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

A: The similarity to the centroid monotonically increases. This is observed from plot2. And table:2.

The increase in similarity is because each cluster becomes tighter with increase in k. The tightness increases because a set of documents which are even slightly different from other set of documents would form a new cluster given an option to do so. Eventually if we keep on the increasing the cluster count, fewer documents would be present in each cluster and those would be very similar. The rate of increase would vary according to the set of documents. If all the documents are very similar to each other, increasing the cluster count would have small increase in tightness whereas if the documents are not alike then increase in cluster count will relatively increase the tightness more. This is also observed in plot2.

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

A: "***Languages***"

- For k = 10:
  - Doc id – 14437, 14439, 14441. All these documents talk about programs and activities offered at ASU for "Slavic/Russian Languages and Literatures". They are from category "Department of Languages and Literatures".
  - Doc id – 14358, 14355, 14442. These documents are about "Careers in Foreign Languages". They refer to advising and application procedure links. They are from category "Welcome to the Department of Languages and Literatures".
  - Doc id – 20928, 20888, 19929. They are about news of things ASU is doing with respect to languages program and department. They are from category "ASU News".
  - Doc id – 1374, 1399. They talk about degree program in Japanese and Spanish. They are of category "Welcome to the Department of Languages and Literatures – Japanese Program and Spanish Program"
  - Doc id – 1869, 1880. They are about "Russian and East European Studies Center" and talk about their newsletter and mission. They are from category "Critical Languages Institute".
  - Doc id – 16700, 17693, 17748. They talk about collections and databases of foreign languages available at ASU libraries. They are of category "ASU Libraries".
- For k = 3
  - Doc id – 14437, 1374, 14439. The documents talk about programs and activities offered at ASU for Slavic/Russian and Japanese Language. They are from category "Department of Languages and Literatures".
  - Doc id – 20928, 20888, 19929. They are about news of things ASU is doing with respect to languages program and department. They are from category "ASU News".
  - Doc id – 16277, 16700, 17693. They talk about collections and databases of foreign languages available at ASU libraries. They are of category "ASU Libraries".

The clusters for k = 3 are present in for k = 10 as well. The clusters become more specific for higher cluster count. Like Department of Languages is further divided into 3 clusters each talking about different topics (Russian Languages, Japanese and Spanish Languages, and Careers in Foreign Languages). And a cluster of critical languages also got formed.

"**Medic Care**"

- For k = 10
  - Doc id – 20952, 19875, 21023. They talk about ASU efforts to improve health care. They are of category "ASU news".

- Doc id – 20590, 21011, 20757. They are also news about ASU's effort to improve health care. They are also of "ASU News" category.
- Doc id – 24945, 24947. Refers to the resources provided by ASU for faculty, student, staff and parents to manage work and family life. Category "Working at ASU".
- Doc id – 23559, 23562, 23574. They talk about the family and child services like child care, elder care and reading room provided by ASU .Category "ASU Child and Family Services".
- Doc id – 23577, 23571, 23569. Child and Family service at different campus (Tempe, West) and deadline to apply for subsidy. Category "ASU Child and Family Services".
- Doc id – 23333, 23365, 23367. They are articles, patient's rights and responsibilities about the service provided on campus. Category "Campus Health Service".

- For k = 3
  - Doc id – 20952, 19875, 20590. They talk about ASU efforts to improve health care. They are of category "ASU news".
  - Doc id – 23565, 23566, 23559. They talk about the family and child services like child care, financial assistance by ASU and related FAQ .Category "ASU Child and Family Services".
  - Doc id – 22816, 4432, 21544. Talk about response to child and dependent care task force, flexible health spending accounts, and Arizona achievement in medical and health care.

As can be observed, for k = 10 some clusters formed are similar to another clusters like the ASU News clusters. But also formed some good clusters which were not present for k = 3 like Campus Health Service, provided distinction between different ASU Child and Family Services, Resources for ASU staff etc. These clusters are not formed for k = 3.

So, as we have seen that with increase in value of 'k' the clusters thus formed get more specific but for very high value of 'k' different clusters formed would contain similar documents, which are not good results. The good value of 'k' would depend on the query. If the query is very generic then high value of 'k' would give good specific clusters whereas if the query is very specific then for higher 'k' value the different clusters will contain similar documents.

(d) 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.

A: Yes, most of the clusters observed correspond to the natural category of the page. With higher value of 'k' different clusters are formed for the documents in same natural category. These clusters formed may point to more specific topics within the main category. But for very high value of 'k', difference between different clusters of same natural category will be very vague/ minimal. (Analysis based on the results mentioned in the previous answer.

GUI:

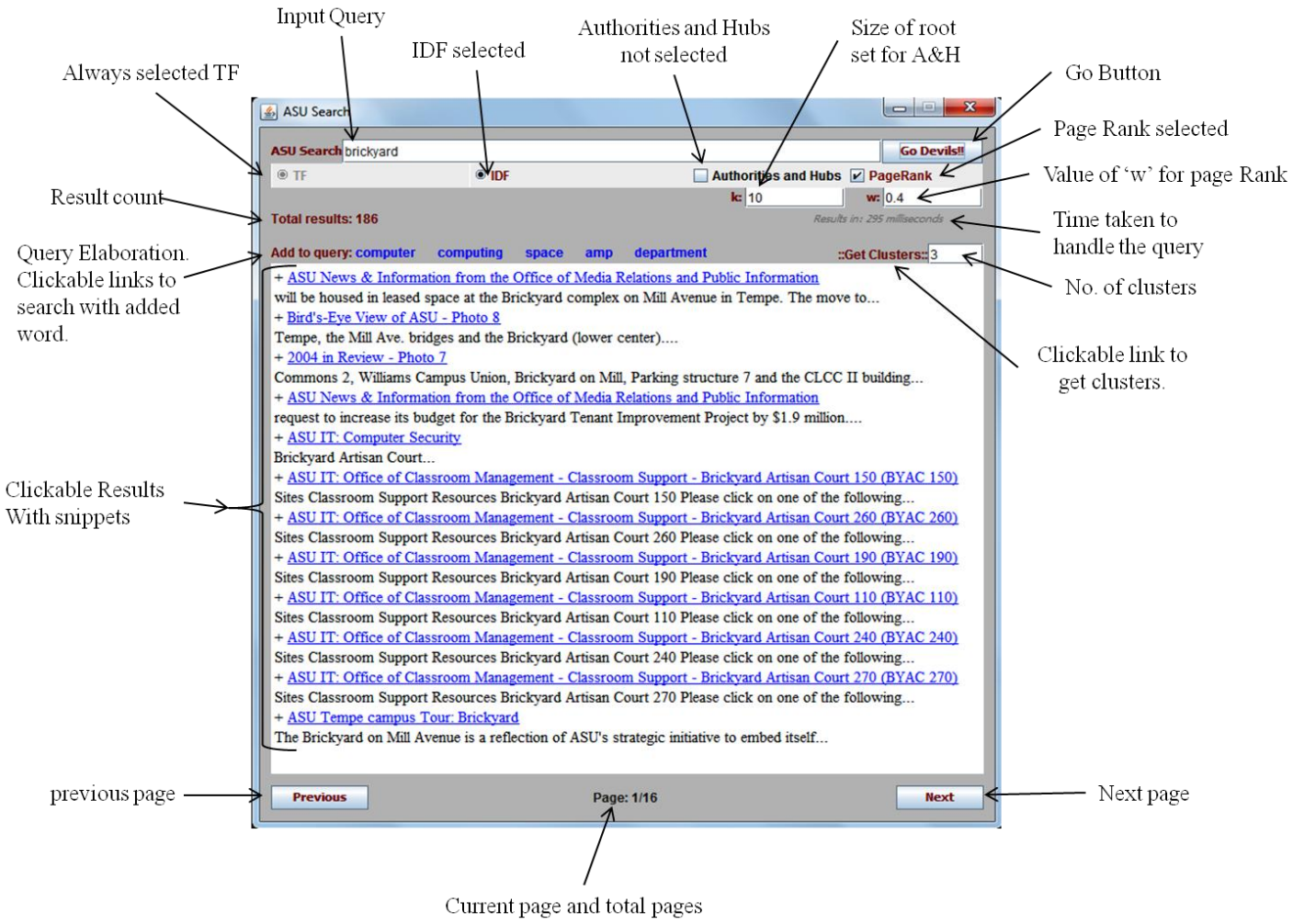


Image1: Show the overall GUI.

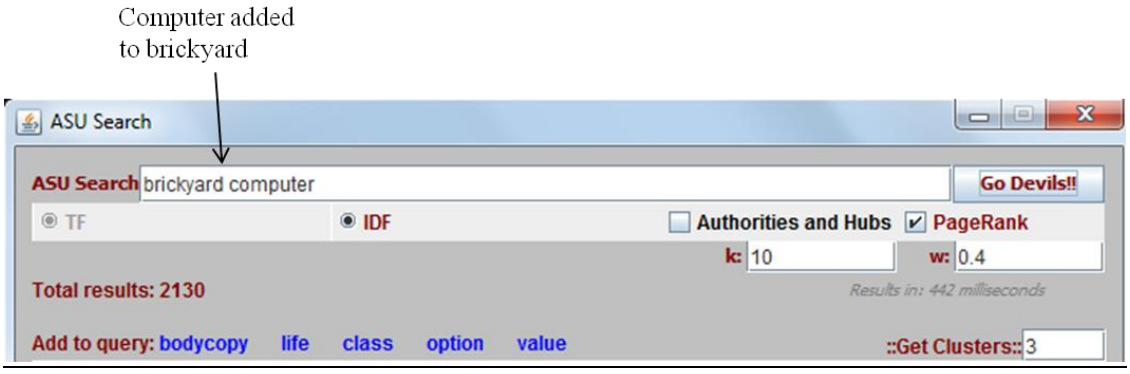


Image2: Clickable links for query elaboration

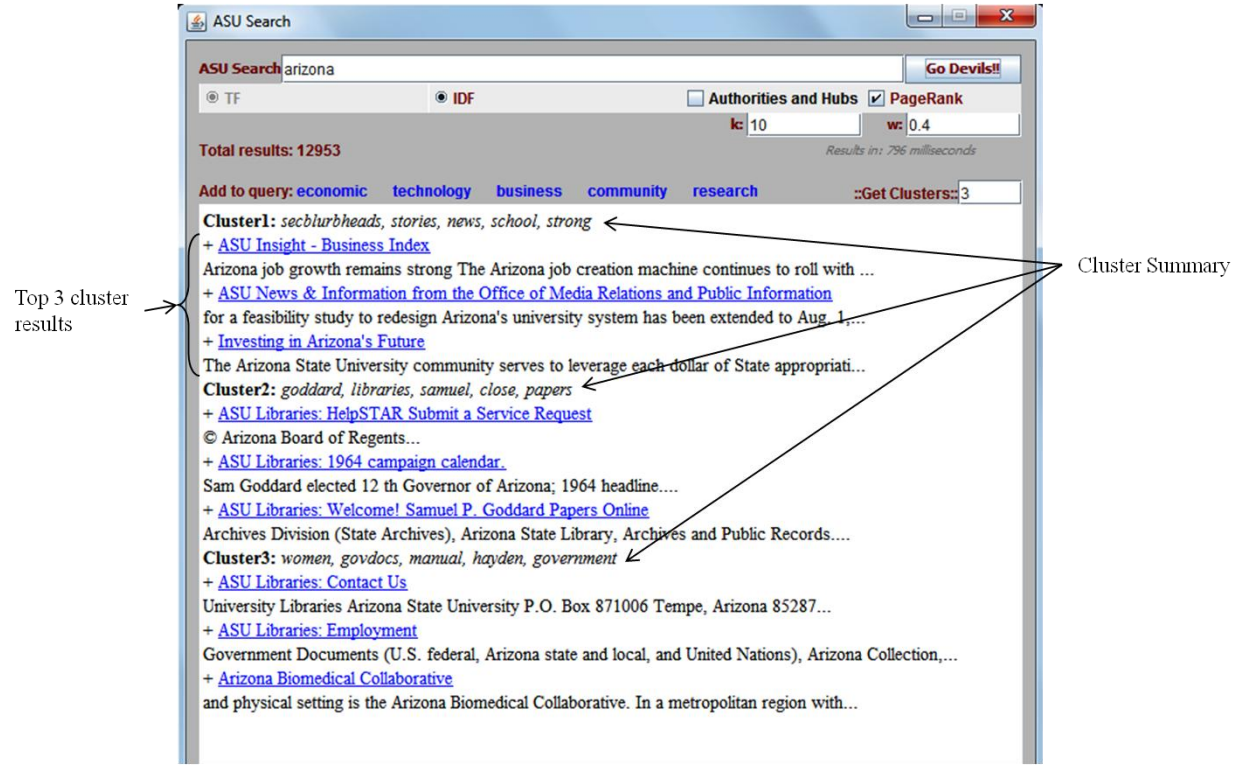


Image3: Cluster results with summary

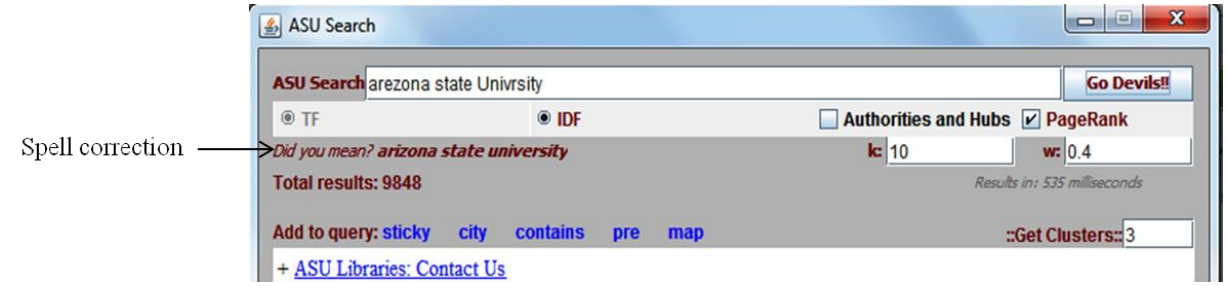


Image4: Tolerant Dictionary (corrects the spelling error)

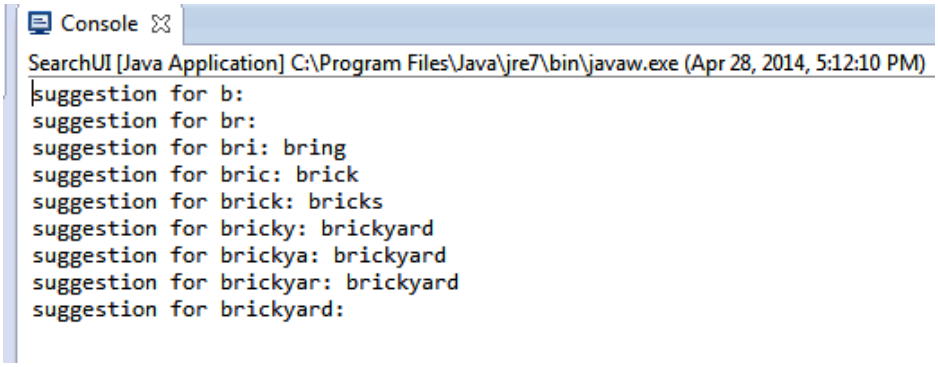


Image5: Showing backend support for query completion.

Snippet generation:

- Time taken: 600-700 milliseconds for document rendered on one page (count 12).
- At query time for each document, the document is parsed using jsoup library. And then query term is searched in various tags of the documents to get the snippet.
- The preference order is:
    - 'description' tag in meta.
    - 'p' tag with no attributes.
    - 'p' tag with attributes.
    - 'li' tag.
    - 'body' tab.
- The snippets generated according to the top preference tags are relevant and good because they describe the main content of the document. Most of the documents successfully get snippets from 'description' and 'p' (with attributes) tags. The ones which don't they have moderate relevance.

Tolerant dictionary/Spell Corrector:

Tolerant dictionary using k-gram Jaccard similarity and IDF is implemented. '*SpellCorrector.java*' implements this functionality. It is an executable application and based on value of "k", it creates a '*spellIndex*' and '*indexMap*' and stores them in '*spellIndex.txt*' and '*termIndex.txt*' respectively. The '*spellIndex*' contains all the k-gram strings and set of index of the terms which contain this k-gram string. The '*indexMap*' gives the term corresponding to an index in '*spellIndex*'. A begin character '*#*' and end character '*$*' is also used for each term.
When the search engine application is launched these files are read and respective indexes are populated. This takes around 500 milliseconds on my machine. When a query is entered, the 'begin' and 'end' characters are added to the query term. For each k-gram string of this query term the set of possible corpus terms is populated and then similarity for all these terms is computed as follows:

$$\text{Similarty}(q, t_i) = (\text{Jaccard similarity})/(1 + \text{IDF}(t_i)) \text{ , where Jaccard similarity} = (\text{union/intersection})$$

The term with the highest similarity is returned as the corrected word.
Interesting queries which can be tried: 'arezona', 'arzona', 'univerity'.

Query Completion:

The same concept and index created for tolerant dictionary is used to do query completion. The end charater '*$*' is not added for this purpose because the word is still incomplete and there is no restriction on the size of the suggested word.

Performance log:

```
Documents norms and IDF computed in: 8977 milliseconds
Page Rank computed in: 9 seconds
SpellCorrector Indexed in: 259 milliseconds
Input Query: computer

Scalar Clusters: computing, related, provide, service, staff,
Scalar Clustering in: 700 milliseconds
Total results for "computer": 2019
Time taken to handle the query: 710 milliseconds.
Snippet generation time: 897 milliseconds.
K-Mean Clustering time: 3085 milliseconds.
```