YIOOP! INTRODUCING AUTOSUGGEST AND SPELL CHECK

CS297 REPORT

Submitted to

Dr. Chris Pollett

By

Sandhya Vissapragada

1. INTRODUCTION

This CS297/CS298 project aims to incorporate the features of autosuggest and spell check suggestions to the queries in Yioop, a PHP-based search engine. These features would help the user to reduce the typing, catch spelling errors or repeat a search. Popular commercial search engines search large indices or popular query lists in under a few milliseconds so that the user sees results pop up while typing. Efficient storage of indices on many servers allows for these minimum response times. Yioop runs on three machines for now. This project aims to implement these functionalities without increasing any load on the servers in turn minimizing the response times for the queries.

As a part of CS297, there were three main deliverables that helped me gain insight into Yioop search engine and autosuggestion techniques. These deliverables would help me in implementing the feature in my final CS298 project.

This CS297 report provides details about the various activities and experiments performed in order to prepare for this CS298 project. One initial task was to install and experiment with Yioop. Then, the next deliverable was to construct a trie with the English dictionary words that can be used for fetching words for autosuggestion. After this, I conducted some experiments with the Google autosuggest feature to see if I could understand how it works. Finally, the task was to incorporate an autosuggest feature in Yioop which could work for popular English dictionary words. Each section in the report explains each of the deliverables achieved in detail. The last two sections of the CS297 report contain the conclusion and references.

2. OVERVIEW OF DELIVERABLES

2.1 Deliverable-1

The aim of this deliverable was to construct a data structure for English dictionary words that can be used to auto complete the words while a user starts typing in Yioop. After researching, it was found that the suitable data structure for this purpose is a trie. [1] A PHP program has been written to serve the purpose that does the following -

- Creates a trie in which words are stored using multi-level PHP arrays.
- The trie is then JSON encoded and a gzip version is outputted.
- It eliminates any words with less than 3 letters or stop words or any words which has non-ASCII characters
- The final gzipped file is around 250KB, which is a reasonable size to send over network and load while Yioop website is loaded [2]

The trie that is constructed will be stored on the Yioop server and will be loaded whenever a user accesses Yioop website. It is further processed using Javascript on the client machine.

Further, timing tests were conducted to see how the addition of trie will effect the page load time of Yioop website. This was monitored using Firefox Web Console option. Loading a JSON trie of size 2.5MB took around 2.5 seconds. Later, the gzip option of HTTP was enabled. It was seen that HTTP gzips the 2.5 MB JSON encoded trie and loads in around 400ms, which is far less than loading a trie directly. The already zipped file, which is about 250KB, followed by decompression can be done in 35ms as shown in Figure 1.

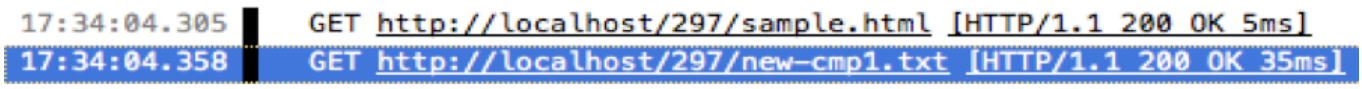

Figure 1: Load time of zipped trie with Deflate option

The third option was to compress the trie with a .gz extension and modify HTTP option to 'gzip,deflate'. By providing this option, the browser expects a compressed file and decompresses it on the fly. This takes just 3 ms to load the trie and the browser automatically decompresses and makes the JSON trie available for autosuggest (See Figure 2)

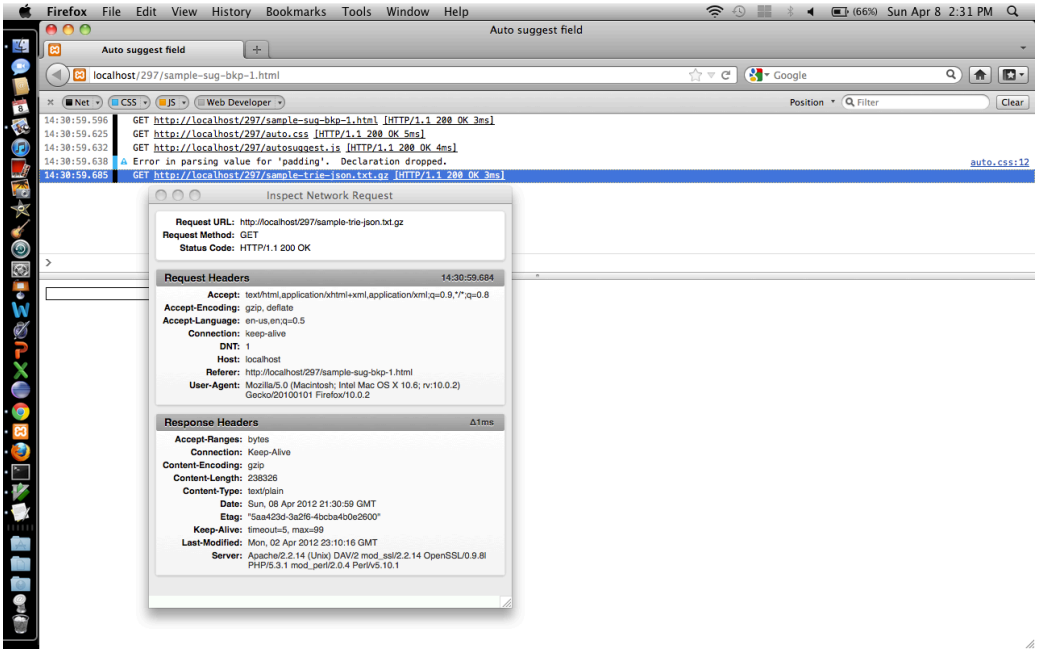

Figure 2: Load time of .gz trie with browser enabled to accept gzip files

After conducting these experiments, it was concluded that

- A compressed trie with gz extension will be made available on Yioop server
- The browser shall be made to accept gzip compressed files
- The browser will unzip the data and it will be used for autosuggest.

2.2 Deliverable-2

This deliverable aimed to examine and analyze how Google's autosuggest functionality works. The Activity Window in the Safari web browser was used as the tool. A series of experiments were conducted on Google search page with various kinds of inputs. The actions going on in the back end were then examined using the Safari Activity window.

The activity window display on opening a Google search page looks some thing like Figure 3.

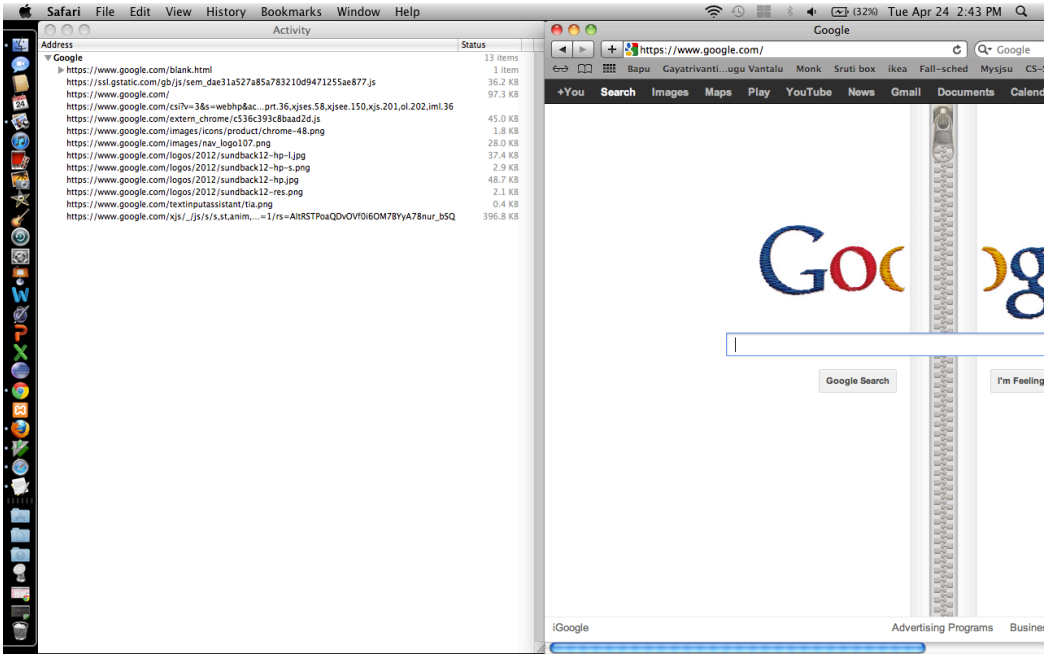

Figure 3: Google search page being inspected in Safari activity window

The data loaded after every click can be downloaded using the activity window. The downloaded file has JSON data with the URL in hex format and the URL contains all the suggested values for the query. The file downloaded for the search query "Soc" can be seen in Figure 4.

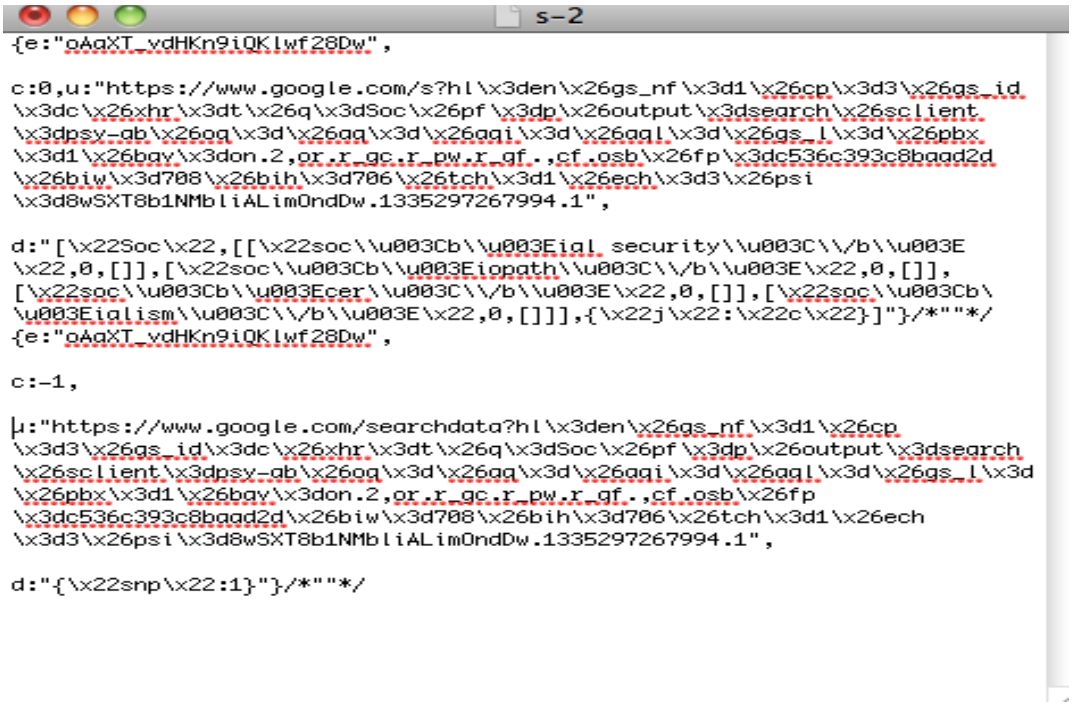

Figure 4: Google search page being inspected in Safari activity window for query “Soc”

The set of urls in the activity window after every letter is entered in "Soc"

https://www.google.com/s?hl=en&gs_nf=1&cp=1&gs_id=3&xhr=t&q=S&pf=p&output=search&sclient=psy-ab&oq=&aq=&aqi=&aql=&gs_l=&pbx=1&bav=on.2,or.r_gc.r_pw.r_qf.,cf.osb&fp=c536c393c8baad2d&biw=708&bih=706&tch=1&ech=1&psi=8wSXT8b1NMbliALimOndDw.1335297267994.1

https://www.google.com/s?hl=en&gs_nf=1&cp=2&gs_id=8&xhr=t&q=So&pf=p&output=search&sclient=psy-ab&oq=&aq=&aqi=&aql=&gs_l=&pbx=1&bav=on.2,or.r_gc.r_pw.r_qf.,cf.osb&fp=c536c393c8baad2d&biw=708&bih=706&tch=1&ech=2&psi=8wSXT8b1NMbliALimOndDw.1335297267994.1

https://www.google.com/s?hl=en&gs_nf=1&cp=3&gs_id=c&xhr=t&q=Soc&pf=p&output=search&sclient=psy-ab&oq=&aq=&aqi=&aql=&gs_l=&pbx=1&bav=on.2,or.r_gc.r_pw.r_qf.,cf.osb&fp=c536c393c8baad2d&biw=708&bih=706&tch=1&ech=3&psi=8wSXT8b1NMbliALimOndDw.1335297267994.1

In trying to analyze the urls, I noticed:

- nf - is always 1
- cp - gives numbering to the urls 1,2,3...
- gs_id - Should be Google search Id
- tch - is always 1
- ech - is also numbering but it will be same if a same query is entered twice during one search

Figure 5 below shows the highlighted part where we can see the words that are retrieved for the autosuggest list

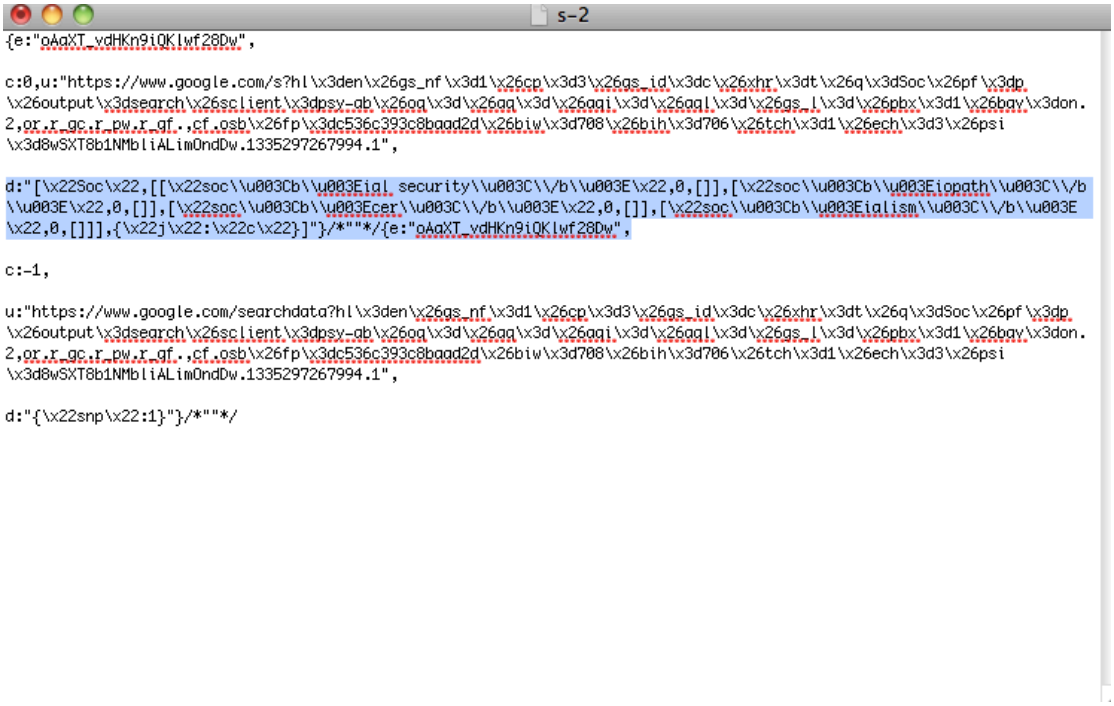

Figure 5: Downloaded file with data on autosuggest results for “Soc”

As we further type the query, the results will be displayed using the top most suggested word. A similar file to the above can be downloaded that has the styling and content of the result page. I was not able to reverse engineer the autosuggest process but understood the sequence of events occur during search.

2.3 Deliverable-3

The objective of this deliverable was to incorporate the autosuggest functionality of English dictionary words in Yioop. Using the .gz file of JSON trie constructed in the previous deliverables, Javascript was written to extract the words for autosuggestion in Yioop

The trie is :

- Loaded while the website is launched
- Every time a key is up in the search box, words are retrieved and displayed
- Only top 6 words are displayed
- The user can hover the cursor on the autosuggestion results and click one of them to place in the search box and works only for the first word.

A high frequency word list was used as dictionary, which was obtained from [3]. After incorporating autosuggest into Yioop, it looked something like Figure 6. This image shows the dropdown for single character entry 'c'.

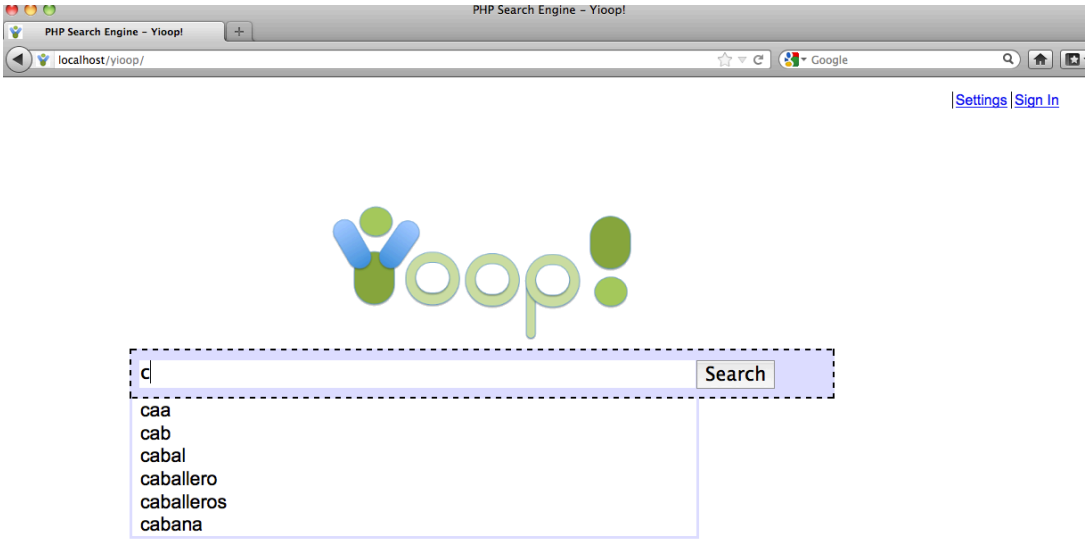

Figure 6: Yioop suggest for character ‘c’

For a search term of more than one character, it looks something like Figure 7.

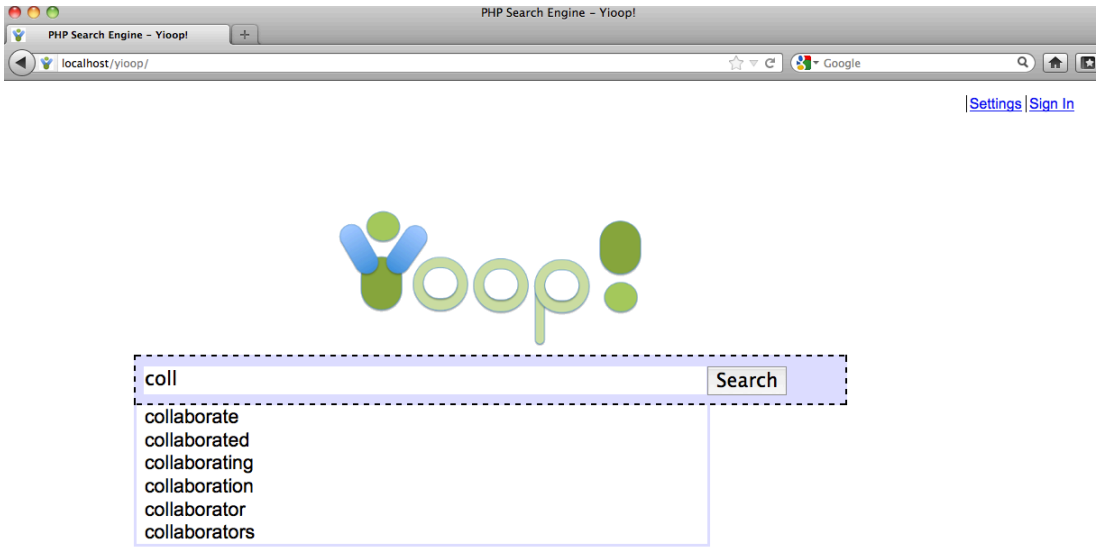

Figure 7: Yioop suggest for word 'coll'

After the intended word is seen in the dropdown, the user can click it to select it. See Figure 8.

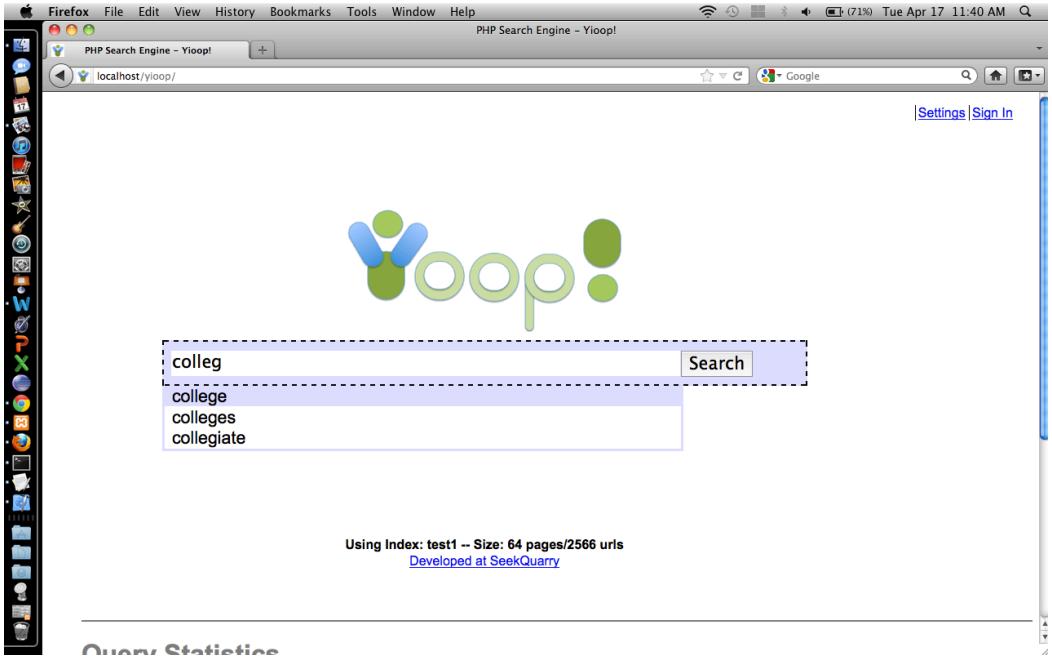

Figure 6: User selecting a word from autosuggest list

2.4 Deliverable-4

The aim of this deliverable was to introduce few more features to the autosuggest functionality implemented in deliverable-3. They include:

- Multi-word suggest feature
- Making the up and down arrows functional to go through the suggested word list
- Introduction of scroll bar if number of suggested words are more than five.

Figure 7 shows an example of multi-word suggestion. Whenever a space is typed in a query, the suggest list appends the previous words of the query for the later suggestions. This way possible phrases are suggested.

Figure 7: Multi-word suggestion for query “Social netw”

Figure 8 shows that the scroll bar appears when there are more number of suggestions.

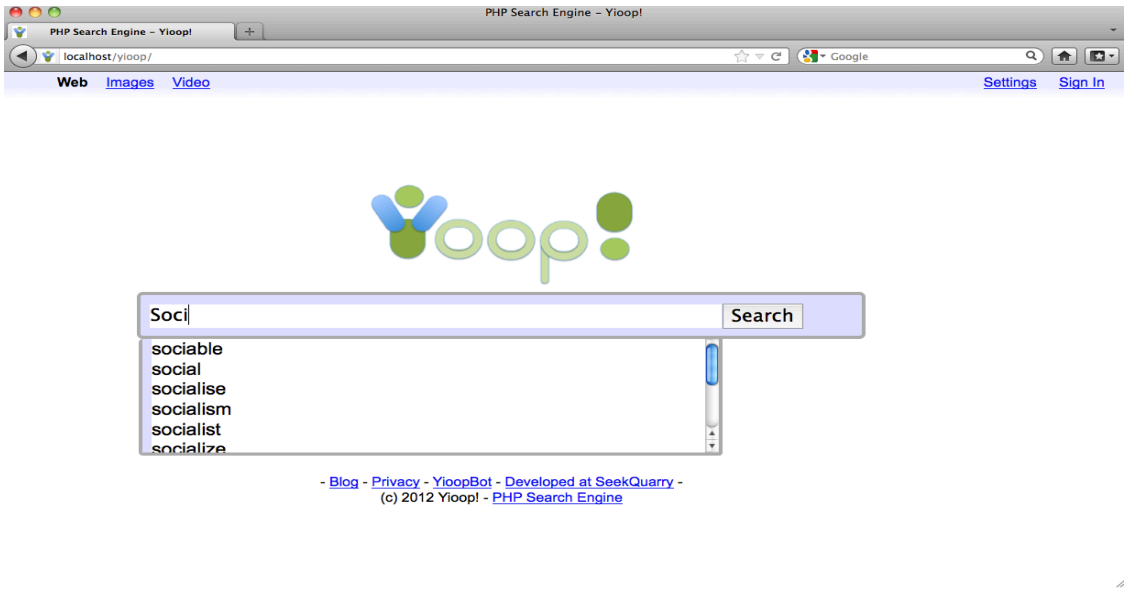

Figure 8: Scroll bar appears in case of more suggestions

Figure 9 shows that using the arrow keys, a word from the suggestion list can be selected and placed in the search box. Along with this feature, cursor hover has been updated to give the control of selected word from the list, to either the mouse or the arrow keys.

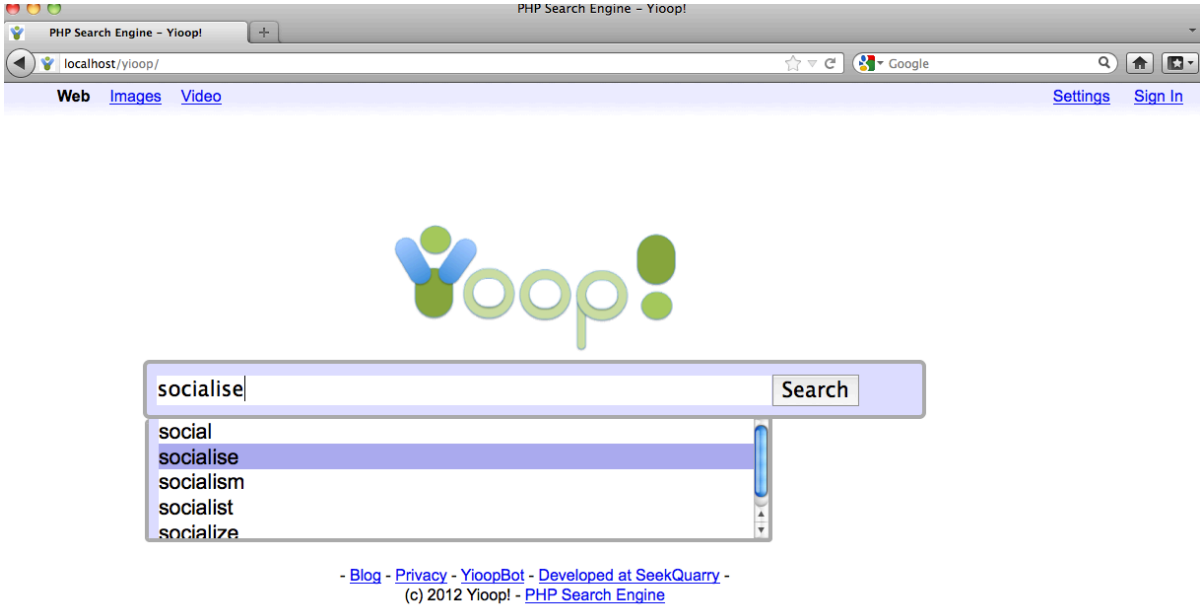

Figure 9: Usage of up/down arrow keys on the list

3. CONCLUSION

In this semester, I studied the basic functioning of autosuggest feature and implemented part of this feature in Yioop. This gave me an insight into the implementation techniques of this feature and also coding for Yioop.

This autosuggest functionality in Yioop will be improved and additional features will be added as a part of CS298. Additional features will include enabling autosuggest for non-ASCII characters, foreign languages, usage of previous user behavior for suggestion and also support for cross-character set input.

4. REFERENCES

[1] Trie: http://en.wikipedia.org/wiki/Trie Retrieved May 15, 2012

[2] Speeding Up Your Web Site: http://developer.yahoo.com/performance/rules.html Retrieved May 15, 2012

[3] Popular English words: http://books.google.com/ngrams/datasets Retrieved May 15, 2012

[4] W3schools: http://w3schools.com Retrieved May 15, 2012