HW5 REPORT - NBCNews

1) STEPS FOLLOWED TO COMPLETE THE ASSIGNMENT

- Built this HW atop the HW4 module which already had the facility of searching and producing top 10 results from files indexed in SOLR using *Pagerank and Lucene's default* indexing algorithm.

- For implementing **spellcheck**, an external spell corrector algorithm was downloaded in php, namely Norwig's Spellcheck Algorithm.

- This algorithm took an external dictionary of words, *big.txt,* to calculate edit distances and appropriately correct the spellings according to it.

- This *big.txt* was created using a JAVA program and the *Apache Tika Parser external JAR,* by scanning and parsing all the html files and create a lexicon which was used for spell correction.

- Now the spellcheck feature was implemented using the lexicon created in PHP.

- When Spell Corrector algorithm was run for the first time a serialized dictionary was made which was used for spellcheck.

- Now whenever, the user input in the front end used to differ from what was already present in the lexicon created in the above steps, a corrected spelling was suggested and the user is shown the results for the correct spelling along with the link to show results for the query user instead. Eg: QUERY : snpachat
  Showing results for *snapchat*
  Search instead for *snpachat*

- For implementing **autosuggest** feature, Solr's inherent feature of suggesting along with AJAX was used to produce the suggestions and shown at real time to the user as the user typed.

- *SolrConfig.xml* was modified to include the *solr.SuggestComponent* and its attributes were also added. In order to handle it along with other requests a *request handler, namely solr.SearchHandler,* was also added to the xml file.

- Now in order to show suggestions to the user as he typed JAVASCRIPT along with AJAX was used to send the query entered up till now to solr's search component and using fuzzy lookup factory and prefix matching solr used to return list of suggestions to the user starting with the same prefix which were then rendered to the UI as a dropdown list for the user to select.

- It assumed that what we send the suggest.query parameter as the beginning of the suggestion. It matched terms in my index starting with the provided characters. So if the query is "br" it will return all the words starting with "br", e.g. "brake" and "brexit" etc.

- For implementing **snippet** generation the query term entered [fully and partial] was matched to produce the snippets.

- The query term was matched in the title, description and then the <body> <p> tags to find the appropriate snippets and then they were displayed.

- If no snippet is found among the above mentioned places, then no snippet is generated.

2) ANALYSIS OF RESULTS

AUTO-SUGGEST EXAMPLES:

SNO	INPUT	AUTOSUGGESTIONS
1	snap	snap, snaps, snappy, snappytv , snapchat …
2	brex	brexit, brexits, brexiteer, brexiteers, brexituk …
3	ill	Illegal, illegally, Illinois ,illness ,illicit…
4	russi	russia, russians, russianembassy , Russian, russiaus…
5	goog	google , googleplus , googlebot, googling, googly

SPELL CORRECTION EXAMPLES :

SNO	QUERY	CORRECTED SPELL
1	Snpachat	Snapchat
2	Brxit	Brexit
3	Donlda	Donald
4	Rusisa	Russia
5	Nsaa	nasa