WEB SEARCH REPORT

Wikipedia PageRank implementation

Umair Naveed, Jesus Zarate, Greg Anderson

ABSTRACT

Used Spark to load and cache large data sets containing Wikipedia pages in XML format. Parsed the data into HTML, plain text, and outgoing links. Used this information to find pages that contained the search keyword along with computing PageRank using the number of incoming links for a given page. The combination of PageRank score and relevant keywords was used to generate results for a given search term.

DATA COLLECTION & PROCESSING

We are required to collect data from a very large data set of Wikipedia pages in the form of XML files. To best utilize Spark, we used the function `wholeTextFiles` to grab the entire collection as a key/value RDD. In terms of processing the data, we chose to use the Wikiextractor to parse the XML into HTML text. Modifications were made to the parser so that it would take advantage of the distributed nature of Spark.

While the Wikiextractor could produce data in HTML, we made the decision to also keep a copy of the data was devoid of HTML tags or special characters. This would become useful when it came to analyzing word counts in each page. Similarly, we wrote in functionality to create a key/value RDD of page titles to outgoing links.

We found that we reused the parsed data (represented by the variable **`converted`** in our code) in several portions of our code. To take advantage of this, we opted to use

caching on that data.

SEARCH

For the search portion of the project, we used a combination of relevant keywords and PageRank, with more weight given PageRank when returning results. We made the assumption that queried keywords would be case sensitive. Therefore a search for 'brazil' and 'Brazil' would be considered distinct. Another assumption we made was to not support searching for special characters.

To implement PageRank, we had a data structure that mapped page titles to outgoing links. By using `flatMap` and repeated use of `map`/`reduce` we were able to produce a transition matrix containing every page in the data set. The values in the matrix corresponded to links between pages. Using the transition matrix we were able to compute a rank for each page, loosely mirroring that of the PageRank algorithm. The resulting RDD (**`page_rank`** in code) was a key/value pair with a page title key and PageRank score value.

In addition to computing PageRank, we computed results based on the number of times the search keyword was present in the Wikipedia page. We removed results that did not any instance of the search term. This RDD (**`word_counts`** in code) had the key/value of page title for key and a list of word counts. By using the join operation on `page_rank` and `word_counts` we were able to produce an RDD that contained page title for key and a PageRank score as well word counts for the value.

Sorting by PageRank score gave us results that were weighted by PageRank and we decided to limit our results to the top 10. This decision was heavily motivated by the second page of Google search results[1].

REFERENCES

1 - https://xkcd.com/1334/