CS 453: Project 3 - PageRank and Indexing

Project Report

Creepy - Indexer

Travis Hall
trvs.hll@gmail.com

Brittany Thompson
miller317@gmail.com

Bhadresh Patel
bhadresh@wsu.edu

Washington State University Vancouver
October 31, 2010

Abstract

The main goal of this project is to index terms in all of the documents, rank them, and get ready for keyword queries. Both the page ranking and the indexing is done using the Map-Reduce paradigm.

1 Overview

For project 2, we used the Map-Reduce paradigm developed by Google for web crawlers for page ranking and term indexing. We take the documents that have already been tokenized, stopped, and stemmed in order to make a list of all the imperative terms, as well as determine the pages importance using a Google-like page ranking algorithm.

2 PageRank

3 Indexing

After the stopping and stemming is complete, the indexer is then ran to catalog all the terms in every document. The indexer is written in python and uses an inverted list structure similar to the indices in the back of textbooks. I used a nested dictionary to hold the list of terms and with each term is attached a list of pages and the occurrences of that term. On the test documents that I used, the output looks something like this: ..., 'international': '3testPage.txt': 1, '4testPage.txt': 1, '5testPage.txt': 1, 'security': '3testPage.txt': 11, '4testPage.txt': 2, ... where the term is listed along with the page that it is found on and the number of times it appears on that page.

4 MapReduce Indexing

While the previous indexer certainly works when run on a single machine, it runs into trouble when being distributed. Since our document corpus can become quite large, we clearly needed a way to distribute the creation of our indices: this is where MapReduce comes in.

One of the first things that was required for doing MapReduce was some extra processing done to the files. Since the way MapReduce works is to simply stream the file into the mapper's STDIN, we needed to find a way to tag the file with its ID. Fortunately, since all the files had, at this point, been processed and stopped and stemmed, we knew that there would not be any HTML (or XML)-like tags still within the document. The obvious answer to tagging was then to simply introduce a new tag as the first line of the document: <`file=[filename]`>. We also needed to ensure that there's a newline character at the end of each file so that when processing we only had to run a regular expression on each line instead of each word individually. All this is done in `prep_data.rb`

5 Roles

Travis Hall MapReduce indexing

Brittany Thompson Indexing

Bhadresh Patel Basic PageRank, MapReduce PageRank, script to run MapReduce PageRank iteratively until convergence.

6 Test Environment

For testing/production purpose, we set up a machine instance on Amazon EC2. The instance id of the machine is `i-7d1e0d17`. The source code is checked out at `/home/ubuntu/creepy/`.

7 Usage Guide

7.1 PageRank

```
$ python lib/pagerank/PageRank -l linkmap.xml
```

7.2 Indexing