Name:	Edwin Rodriguez (ED04)
ID:	Mustafa Abbasi (D07)
Project:	Mini-Project 2
Class:	CMPUT 291
Due Date:	Nov 27, 2015

No Collaboration was done for this project.

C291G40's Review Database System:

A1. General Overview

The primary purpose of the Information Retrieval System(IRS) is to facilitate searching for user-defined queries using the physical layer. This is done by parsing the input of the system (a single file containing one or many user reviews) in order to simplify the input and eventually create four queryable Berkley DB indices. These indices will then be used as the basis for the user queries allowed by the system.

The IRS is composed of three main phases:

Phase 1: Prompts the user for the review file. Once the file is found, it parses the file and creates four sub-files (reviews.txt, pterms.txt, rterms.txt and scores.txt) containing key information about the reviews.

Phase 2: The IRS sorts the sub-files created in phase 1 and creates four database indices (rw.idx, pt.idx, rt.idx, and sc.idx) using the sorted sub-files.

Phase 3: The IRS will prompt users to enter a query. Once entered, the IRS will try to retrieve all the information that fits the query using the databases indices. Once done, it will prompt the user for another query.

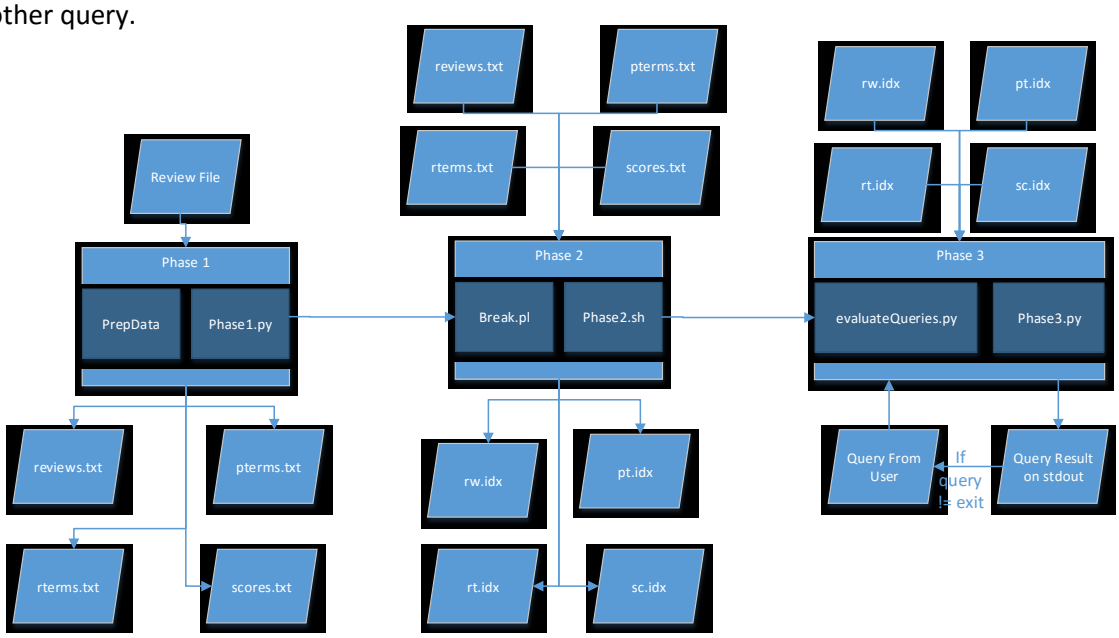

Figure A: Flowchart of the data flow in the Information Retrieval System (IRS)

A2. User Guide

In order to run the IRS, three important scripts need to be executed in the following order:

1. PrepData.py (Phase 1)
    a. This script will prompt the user for the location of the review file.
        i. This will read the review file and create the sub-files (*.txt).
2. Phase2.sh (Phase 2)
    a. This script will delete all existing indices and it will re-create them using the most recent versions of the sub-files (*.txt).

    b. If you are unauthorized to run the script due to Linux permissions, type in the following command: chmod +x phase2.sh.
3. Phase3.py (Phase 3)
    a. This script will endlessly prompt the user for a query.
        i. The query will return nothing if the query is unable to find any information.
        ii. The query will return the index followed by the summary of the review of any hit.
    b. To exit, type exit as a query.

B1. Description of Algorithm Used for Evaluating Queries.

The basic algorithm used to evaluate queries involves:

1. Splitting the query into individual components
2. Comparing each component to special key characters (<, >, …) or words (pprice, rscores, …).
    a. If a component is found, combine the key characters/words with preceding/proceeding components to form a query (What is combined depends on the type of key found).
    b. Compound queries or queries with multiple conditions are broken into simpler individual queries (max only, min only, …).
    c. The contents of wildcard keys were compared with the first few characters of indexed items.
3. Evaluating each individual query was done in order of:
    a. Conditions on indexed data (title, review, score).
    b. Conditions on unindexed data (price, date) on common results of a.
4. Finding the intersection of all the query results (in the case of compound queries with multiple conditions).
5. Fetching the full results from rw.idx using the indices obtained through 1-4.

B2. Analysis of Algorithm Efficiency

The efficiency of the search algorithm depends on two main factors. The first factor involves the searching of the requested key(s). This search comes at a cost of finding an index in a B+ tree structure which is $O(\log_f(N))$. The cost of compound queries come at a cost of multiplying this base cost by the number of conditions in the query. The second factor involves fetching specific contents from the rw.idx hash index file. This comes at a cost of O(1) since the hash usually unique. This means that the overall efficiency of the algorithm is:

$$Efficency = [C*O(\log_f(N))]*[R*O(1)]$$

Where: *C=number of conditions*
f=base of the B+ tree index
N=the size of the database
R=the number of results from the B+ tree queries

C. Testing Strategy

The testing strategy for the IRS involved the following steps:

1. We used a known data set as an input file. This allowed us to know what to expect of a correct query result.

2. We testing each type of query individually. This allowed for the development of the code for each individual type of query.
3. We tested wild cards.
4. We tested compound queries that involved multiple terms and data ranges.

The usual test cases involved the following types of queries:

1. Queries that involved looking for specific/similar items (terms, pterms):
    a. This involved counting and comparing the results with the known data.
2. Queries that involved ranges:
    a. This involved verifying that the results were inside the range of the queries (<max, >min).
3. Queries that involved dates:
    a. Much like the queries for ranges, we verified that the results occurred before/after the queried date (Depending on the query).
4. Queries that involved wild cards:
    a. The results were compared to similar queries that did not use wild cards.
5. Compound queries:
    a. The results were compared to the results of smaller individual queries.

D. Group Work Break-Down Strategy.

The breakdown of the project was explicitly indicated in initial stages of the project (phase 1). The breakdown of the project changed as the development of the IRS progressed however. Work was divided into two streams: Testing/Debugging/Documentation and Development.

The github repo shows the activity of the project and contains some of the test files used to develop the IRS. please see the following link for the location of the github repo:

https://github.com/c291g40/project2

Break-down:

Edwin	Mustafa
Parsing function used for phase 1 (3hrs)	Phase 1 &2 (3Hours)
Testing/Debugging of IRS system (2hrs)	Phase 3 (9 Hours)
Documentation (4 hours)	Testing (1 Hour)
Total: 9 hours	Total: 13 hours

E. Assumptions

It is assumed that the review file will always contain the following fields in this order:

1. product/productId:
2. product/title:
3. product/price:
4. review/userId:
5. review/profileName:
6. review/helpfulness:
7. review/score:
8. review/time:
9. review/summary:
10. review/text: