University City Dublin

MSc in Computer Science(Conversion)

Software Engineering Project

Metasearch Engine Implementation

Colm Nolan

09793011

colmnolan@gmail.com

Lecturers: Dunnion John, O'Sullivan Tadhg, and Naughton Martina

25/07/2011

Table of Contents

Login and Passwords details

The three main modules can be viewed at

Main Module www.dandysearch.com

Evaluation Module www.dandysearch.com/index_evaluation.php

Data Graphing Evaluation Module www.dandysearch.com/charts/select_charts_category.php

All the the code to generate the above is submitted in the dandysearch_code folder including the data from the evaluation database is SQL form. If you need direct access to the server you can call me on 0879904212.

1 Introduction

This software engineering project en composed 4 main objectives. These were to

- design and build a metasearch engine. The three source engines used was Bing[1], Yahoo[2] and Blekko[3].
- Evaluate the results obtained from the metasearch engine relative to the results obtained from a reference search engine. In this case google[4].
- Analyse and present the data from the evaluation stage.
- Perform a user study on the metasearch engine.

1 http://www.bing.com/developers
2 http://developer.yahoo.com/search/boss/
3 http://blekko.com/
4 http://code.google.com/apis/customsearch/v1/getting_started.html

2 Functional Details of the application

Metasearch Engine Functions

The metasearch engine can display search results in three seperate ways. These are

- Show Search Results for Bing, Yahoo, Blekko and Google in separate lists
- Show a non aggregated list with Bing, Yahoo, and Blekko Search Results merged
- Show an aggregated list with Bing, Yahoo, and Blekko Search Results merged.

It can do preprocessing on the query such as removing stopwords, stemming[5] the query string, and changing the query to a string compatible with multiple boolean searches.

Evaluation Functions

The quality of the non aggregated list search results relative to google. The quality of the aggregated list of search results relative to google. This measure was achieved by using the following metrics for measuring relevance: Precision, Average Precision, Mean Average Precision(MAP) and Precision @ n(10 in this case since most people are only generally interested in the top 10 results). See http://en.wikipedia.org/wiki/Information_retrieval for formal definitions of the formulas above. See below for interpretations of the formulas in relation t this project.

Calculation on Each query

Precision = no snippets appearing in both the aggregated list and google search / total snippets in aggregated list.

Average Precision = sum of precisions at each recall point / total snippets in aggregated list.

Precision at n(10) = no snippets appearing in both the aggregated list and google search in the top 10 / 10.

Calculation on the average of all 50 queries

Mean Average Precision = sum of Average Precisions for the 50 queries/ 50.

Average Precision for All queries = Sum of 50 precisions / 50.

Average Precision at n = sum of 50 Precisions ant n(10) / 50.

A relevance record will also be taken to check MAP calculations, retain a record of the raw data and also allow us visualize the data better. See Appendix A for details.

5 http://www.chuggnutt.com/stemmer-source.php

Evaluation Presentation Functions

This module involved the manipulation of the raw data from the evaluation into graphical representation to illustrate the findings of the results. See Figs 5.1 to 5.8

3 Design

Metasearch Engine Design

The search engine was designed and implemented in an object oriented fashion. Its architecture follows the design pattern model-view-controller[6]. See Fig below.

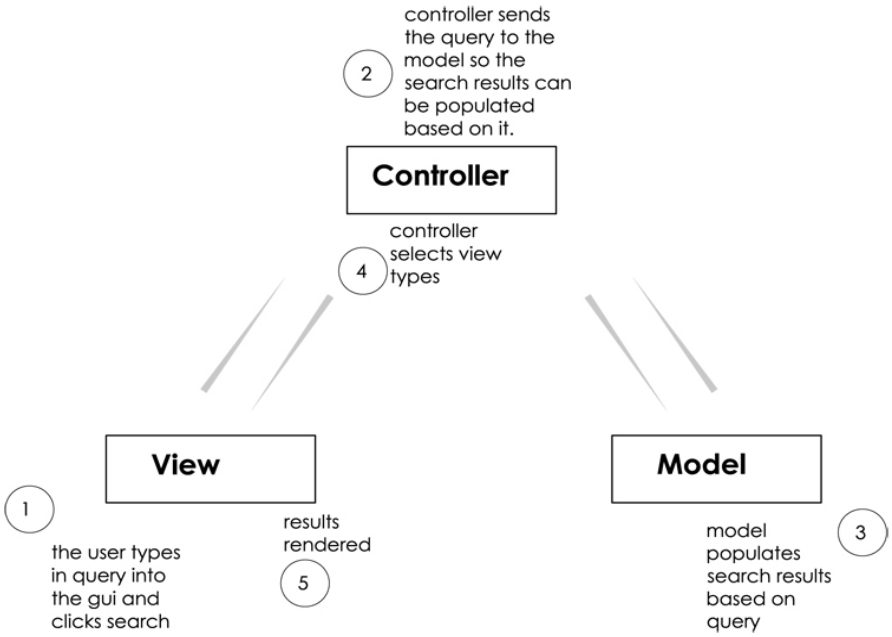

Fig. 3.1 Model-view-controller Architecture

The controller controls all the movement of the data between the view and the model. The instance of the controller object is declared in the index.php file. It is then ran by invoking the method invoke.

Within the controller class instance's of all the other main objects are declared.

The index.php file is located in the public_html. All the controller classes are stored in the folder controller. All the model classes are stored in the model folder. Anything to to do with the front end is stored in the view folder.

See controller Data Structure in Fig 3.2

6 http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

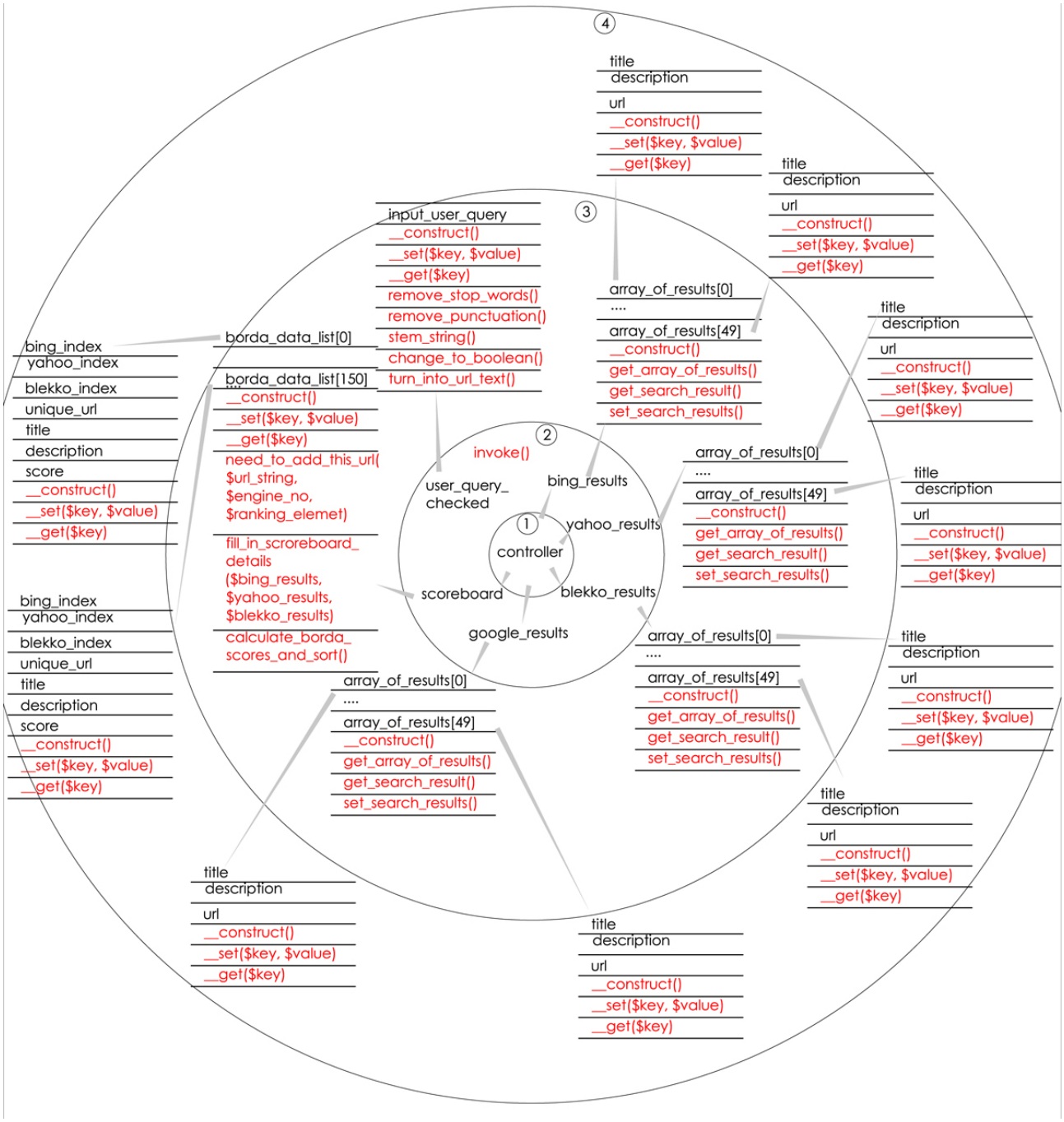

Fig. 3.2 Data Structure for the controller object

The above data structure is the main controller object. It is to be read from the inside out controller->bing_results->array_of_results etc. Note the properties are shown in black and the methods are shown in red. Also note the circles represent all the layers of abstraction, 1 to 4.

Also note where the methods remove_stop_words(), remove_punctuation(),stem_string(), change_to_boolean() and turn_into_url_text() are attached to the user_query_checked object.

Also note the methods where the aggregation list gets populated and aggregated in the object

scoreboard called fill_in_scoreboard_details($bing_results, $yahoo_results, $blekko_results) and calculate_borda_scores_and_sort().

Aggregation Method

The aggregation method picked was the Borda's Rank Aggregation Method based on the Borda count election method[7]:

Borda's method is a "positional"[8] method, in that it assigns a score corresponding to the positions in which a snippet appears within each web searches ranked list of results, and the snippets are sorted by their total score.

Example: query sent into three search engines and the results returned.

Bing Search	**Yahoo Search**	**Blekko Search**	**Points**
Snippet B	Snippet C	Snippet C	2
Snippet C	Snippet B	Snippet A	1
Snippet A	Snippet A	Snippet B	0

Snippet	**Points**	**Position**
Snippet A	1	3
Snippet B	3	2
Snippet C	5	**1**

Fig. 3.3 Borda Aggregation

This aggregation ranking system was picked because of its simplicity and suitability for ranking the snippets. Given that each snippet only contains approximately 180 characters(25 words) analysing term frequencies would not yield good results. Each snippet tends to have the same number of occurrences of the query term. The existing ranking in Bing, Yahoo and Blekko is well utilized.

7 http://en.wikipedia.org/wiki/Borda_count

8 J. C. Borda. Mémoire sur les élections au scrutin. Histoire de l'Académie Royale des Sciences, 1781.

GUI Design

The front end was designed to be inviting, user friendly and straight to the point. Its target audience was the general population so it needed to be easy to use.

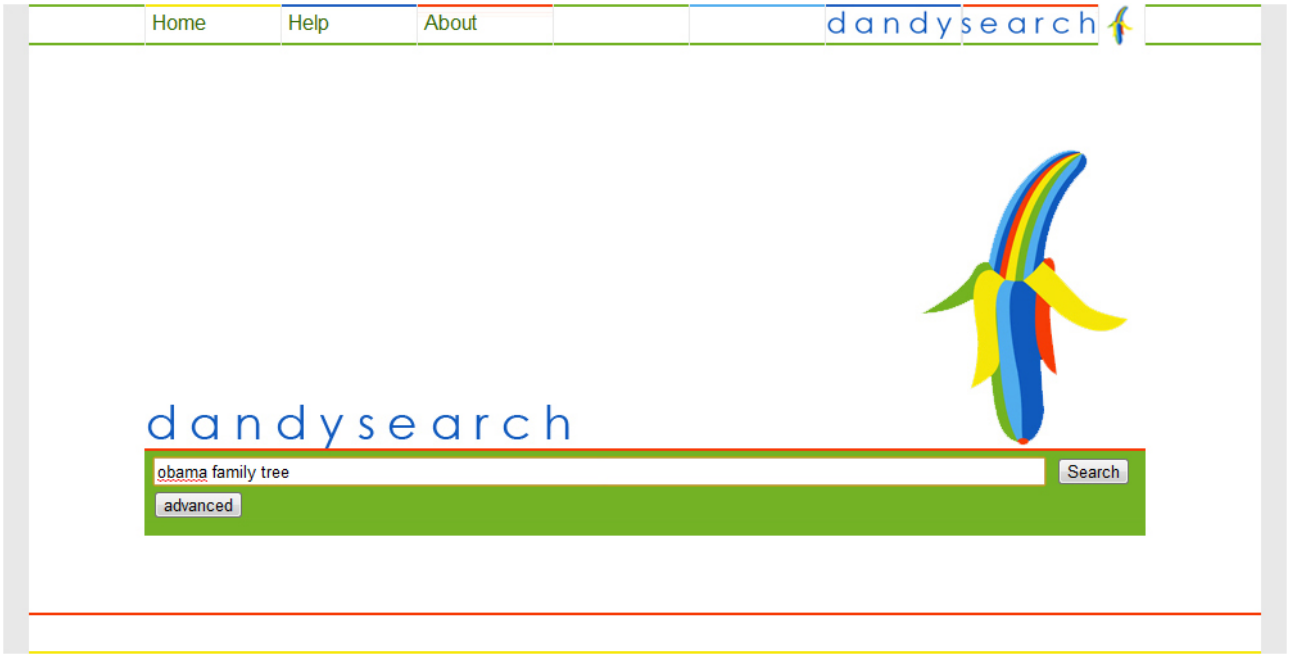

Fig. 3.4 Home Page

This is the default search display on the home page. Any queries here will return an aggregated list of results.

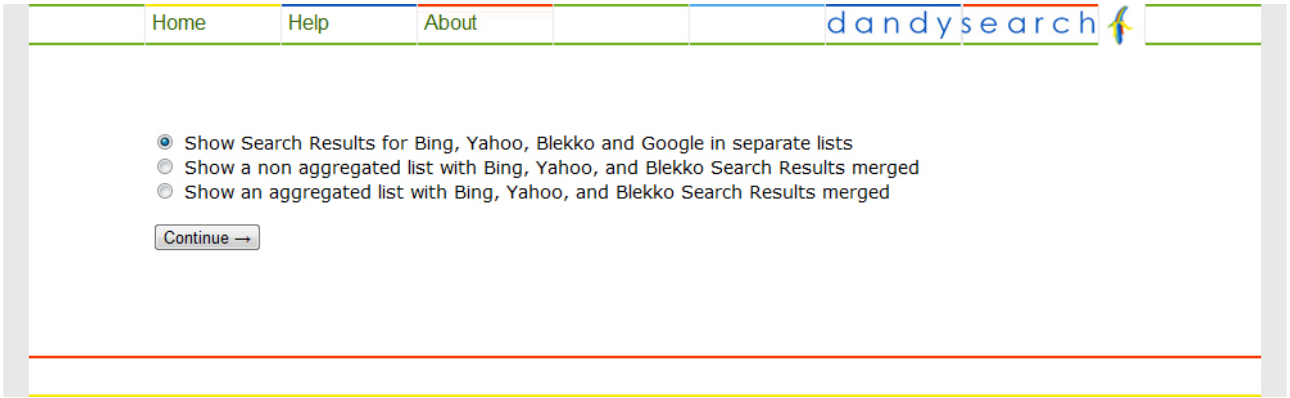

Fig. 3.5 Select Search Results Display Type

If a user clicks advanced he/she will be given the three different display options which can be seen in Fig 3.7, Fig 3.8 and Fig 3.9.

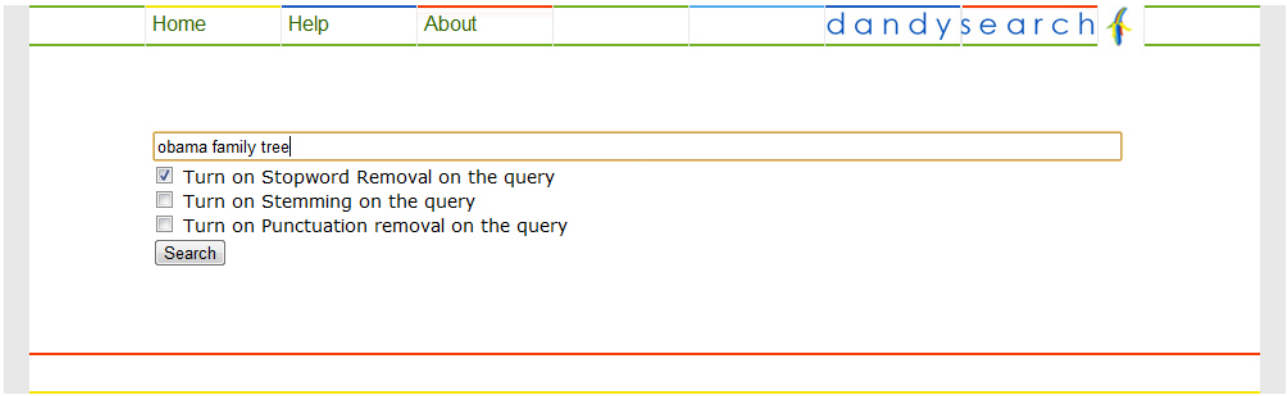

Fig. 3.6 Query Preprocessing Manipulation

This is where the user can decide what type of query preprocessing he/she wants to perform.

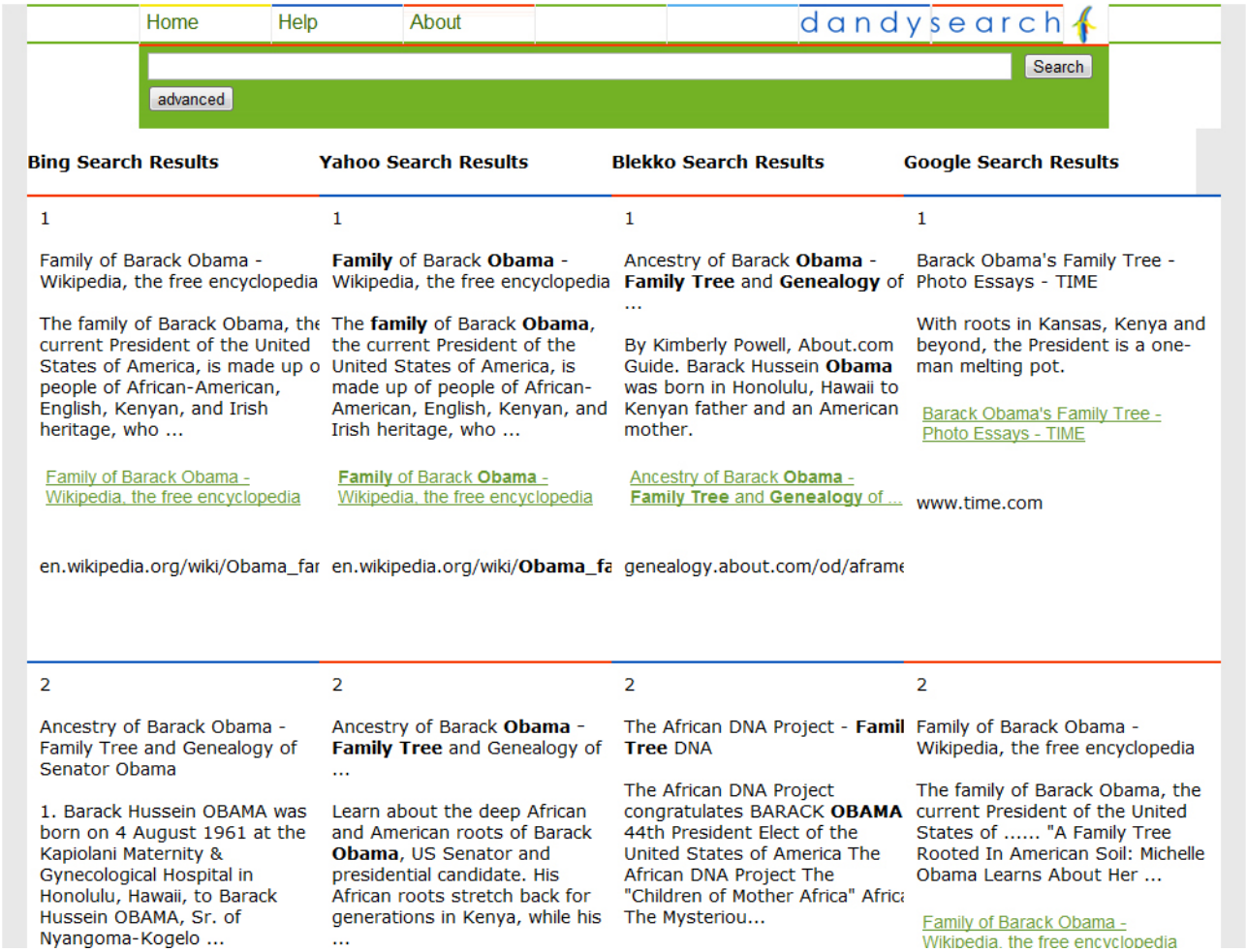

Fig. 3.7 Query Results in 4 lists from the 4 engines

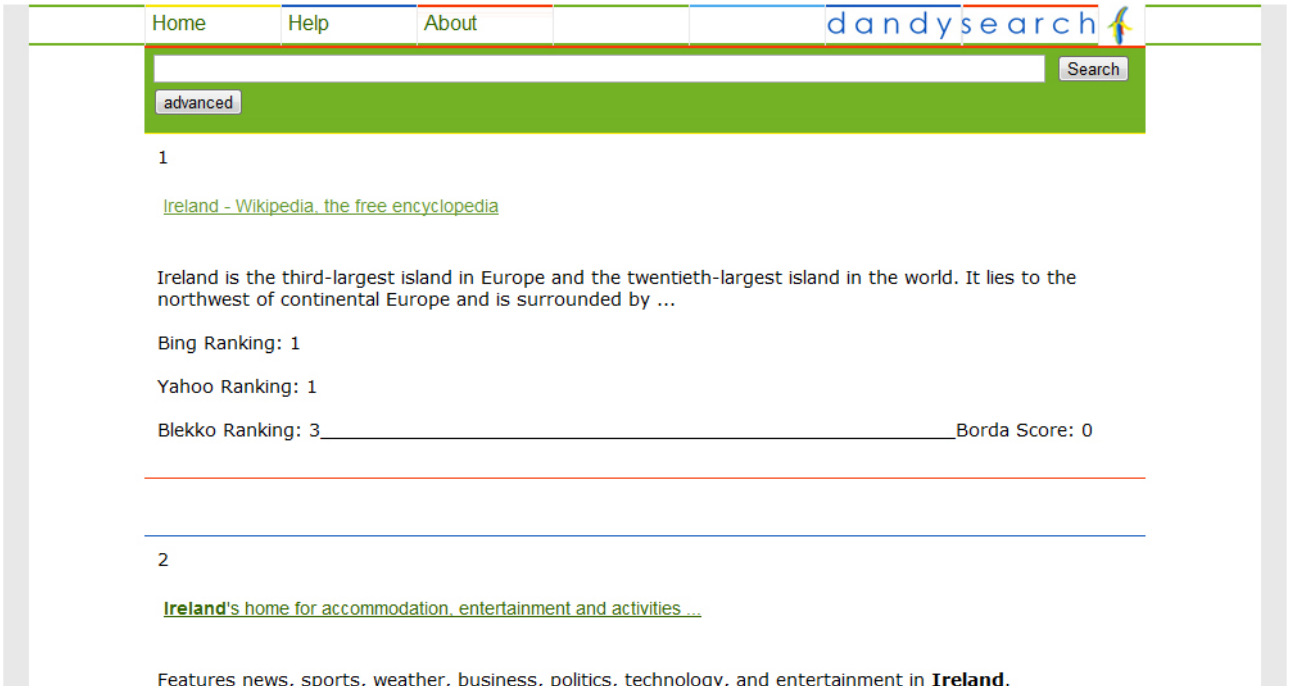

Fig. 3.8 Non aggregated Display

Fig. 3.9 Aggregated Display

Metasearch Engine Evaluation Design

This module worked on the same principal as the last except an extra object was included to evaluate the non aggregated search engine and the aggregated search engine results. A different controller was also used to manipulate the model classes so different types of evaluations could be carried out.

Please see Data Structure for the controller_evaluation object below in Fig 3.10

The results were then saved to the database in their relevant evaluation tables.

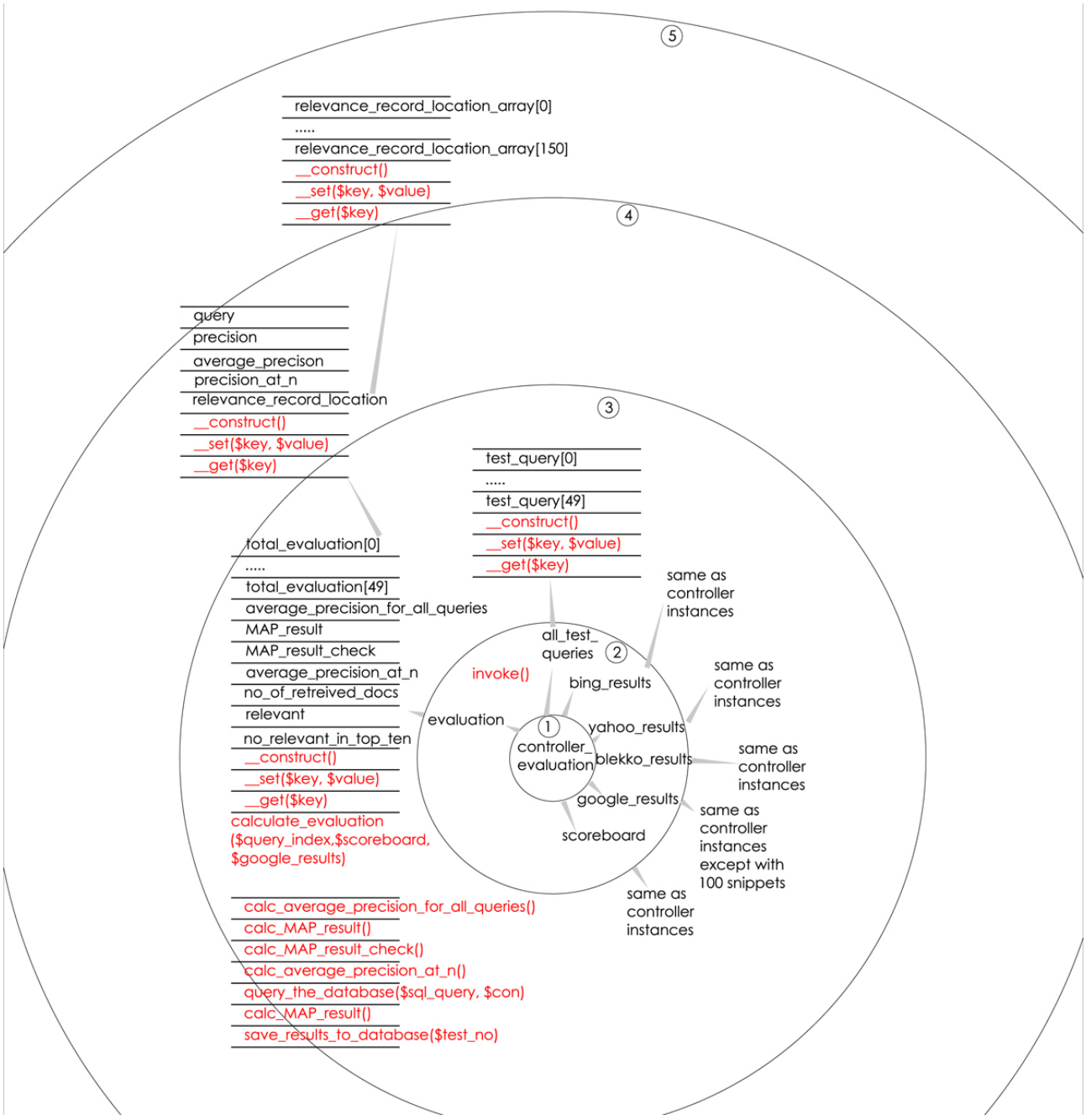

Fig. 3.10 Data Structure for the controller_evaluation object

Fig 3.10 is a representation of the controller object for the evaluation application. Note the methods shown in red which perform the evaluation calculations. In particular note the method calculate_evaluation($query_index, $scoreboard,google_results) which brings in the google object and the scoreboard object for evaluation. This method performs the evaluations on each of the 50 queries.

Fig. 3.11 Evaluation Front End Test Selection

Test 1 would involve testing the Non Aggregated list with no Stemming and no stop word removal using the 50 test trecWebTrackQueries_09.

Test 2 would involve testing the Non Aggregated list with Stemming and stop word removal turned using the 50 test trecWebTrackQueries_09.

Test 3 would involve testing the Aggregated list with no Stemming and no stop word removal using the 50 test trecWebTrackQueries_09.

Test 4 would involve testing the Aggregated list with Stemming and stop word removal turned on using the 50 test trecWebTrackQueries_09.

On completion of the evaluation all the calculations were saved to the database.

Evaluation Presentation Design For Graphs

This was achieved by using dedicated scripts to access the database of evaluation results which would be then presented to the user in a graphical way. All the related files where stored in a dedicated folder called charts. To generate the the graphs the extension jpgraphs was used which makes use of the PHP's basic graphics extension gd2[9]. CSS[10] was also used to present some of the graphs.

4 Technologies Used

The main technologies used were

PHP[11]: The main language used on server side to aggregate and present results.

MySQL[12]: Used to store evaluation details.

HTML[13]: The markup language used to present to the client

Extensible Markup Language (XML)[14]: ordered information received back from Bing and Yahoo.

JavaScript Object Notation (JSON)[15]: ordered information received back from Blekko and Google

Other implicitly used technologies: HTTP, Apache Web Server[16], Internet Explorer, Google Chrome

The application was developed on a Windows Environment[17]. For writing and editing the code for the application I used Gvim[18] along with numerous plugins such as Project, Ctags and Snipmate. For editing the database I used PHPmyadmin[19] with SQL.

9 http://jpgraph.net/
10 http://en.wikipedia.org/wiki/Cascading_Style_Sheets
11 http://www.php.net/
12 http://www.mysql.com/
13 http://en.wikipedia.org/wiki/HTML
14 http://en.wikipedia.org/wiki/Xml
15 http://en.wikipedia.org/wiki/Json
16 http://www.apache.org/
17 http://www.microsoft.com/windows/
18 http://www.vim.org/
19 http://www.phpmyadmin.net/home_page/index.php

5 Evaluation Results and Analysis

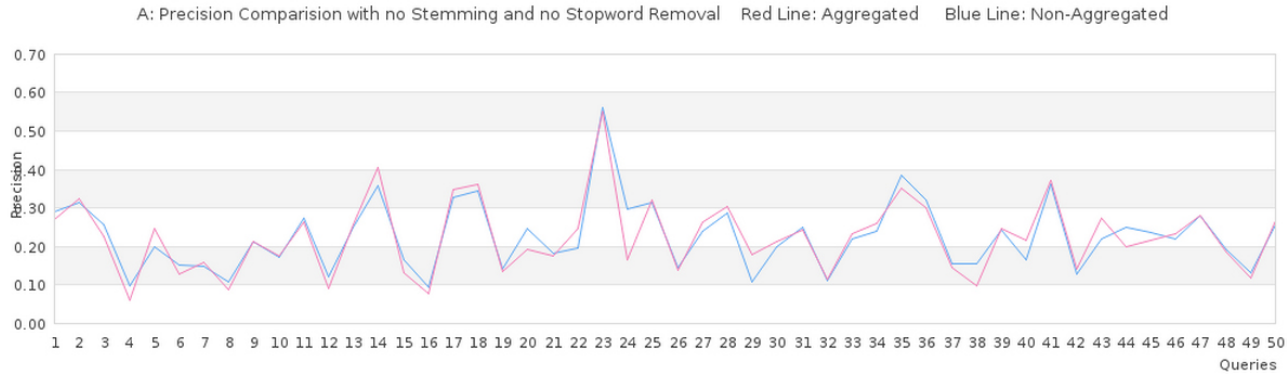

Fig. 5.1 A Precision Comparison(Test 1,3) with no Stemming and no Stopword Removal

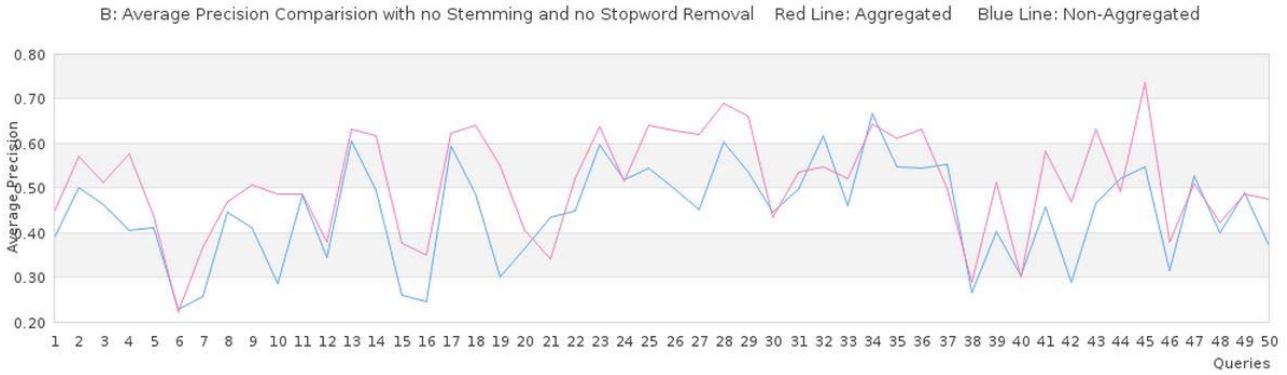

Fig. 5.2 B Average Precision Comparison(Test 1,3) with no Stemming and no Stopword Removal

Fig. 5.3 C Precision At N Comparison(Test 1,3) with no Stemming and no Stopword Removal

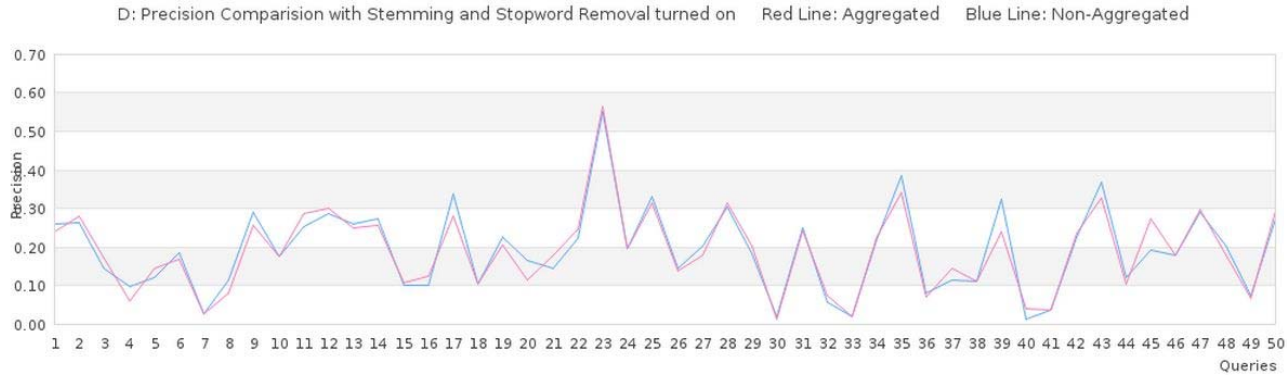

Fig. 5.4 D Precision Comparison(Test 2,4) with Stemming and Stopword Removal turned on

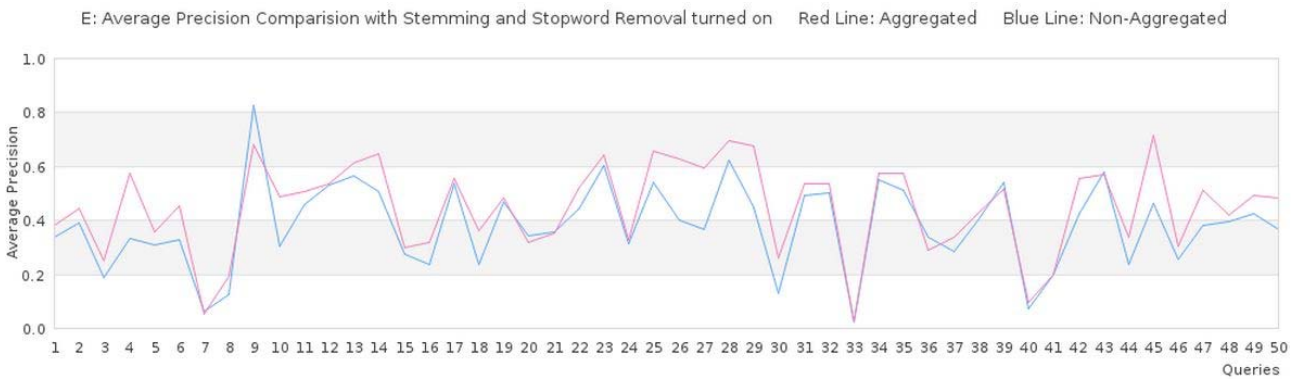

Fig. 5.5 E Average Precision Comparison(Test 2,4) with Stemming and Stopword Removal turned on

Fig. 5.6 F Precision At N Comparison(Test 2,4) with Stemming and Stopword Removal turned on

Test No and Description	**Average Precision For all Queries**	**Mean Average Precision(MAP)**	**Average Precision at n(10)**
1:Non Aggregated, Stopword and stemming off	0.23	0.45	0.53
2:Non Aggregated, Stopword and stemming on	0.19	0.38	0.42
3:Aggregated, Stopword and stemming off	0.22	0.51(6% better than non agg)	0.65 (12%better than non agg)
4:Aggregated, Stopword and stemming on	0.19	0.45(7% better than non agg)	0.53(11 %better than non agg)

Fig. 5.7 Results Comparison

As can be seen in the graphs above the aggregated results give a better rating for Average precision and Precision at n. Precision does not change in either case because of it does not take ranking into consideration.

The graphs can be more clearly seen at http://www.dandysearch.com/charts/plot_precisions.php The aggregation method ranks the documents finishing the top 10 the best. Note Precision at n yields very good results with an improvement of 12%.

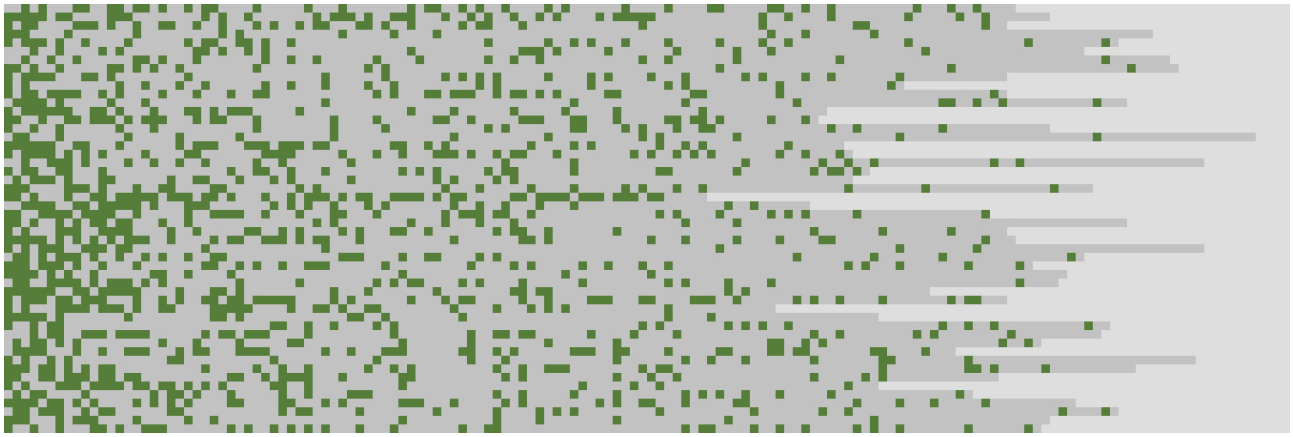

Fig. 5.8 1 Non Aggregated with no Stemming and no Stopword Removal

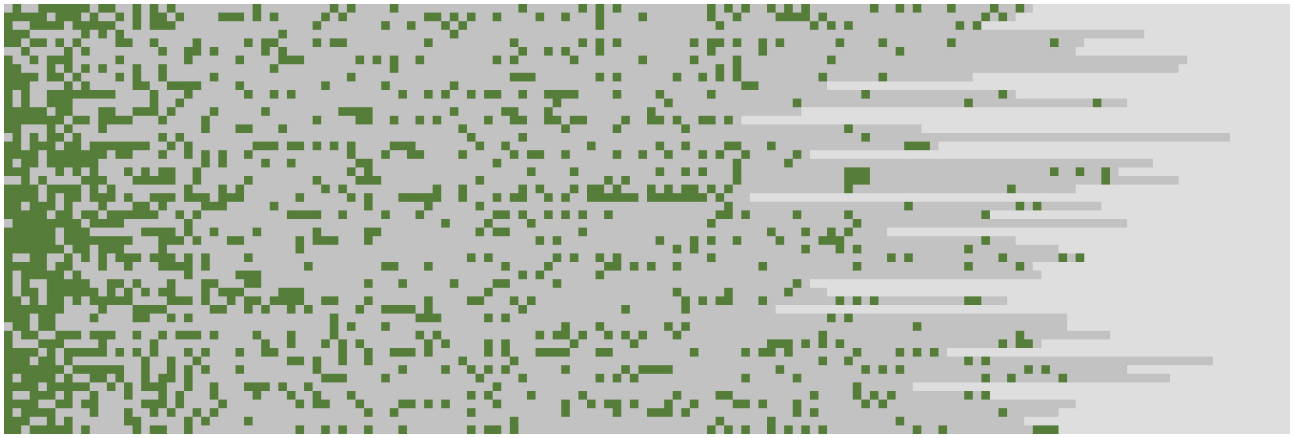

Fig. 5.6 3 Aggregated with no Stemming and no Stopword Removal

This is a visual representation of relevant documents. The x-axis represents the possible 150 unique snippets from the 3 search engines. The y-axis represents all 50 test trecWebTrackQueries_09. The first line is obama family tree, the second french lick resort casino etc. The green squares represent a relevant document while the dark grey represents the non relevent documents. The light grey represents no document. Note how in fig 5.6 more relevent documents push towards the top(left) after aggregation. This can be more clearly seen on http://www.dandysearch.com/charts/print_divs.php?test_no=1

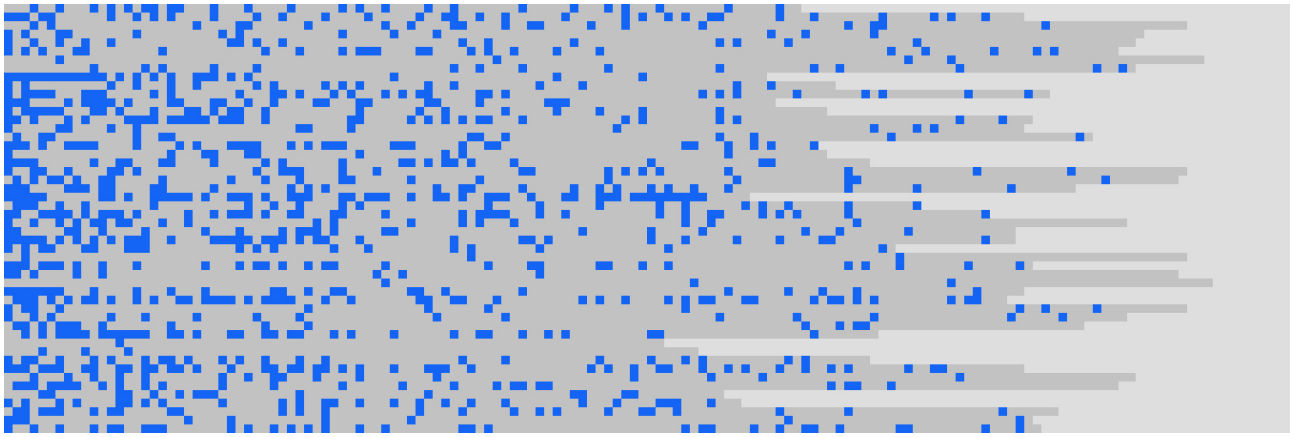

Fig. 5.7 2 Non Aggregated with Stemming and Stopword Removal turned on

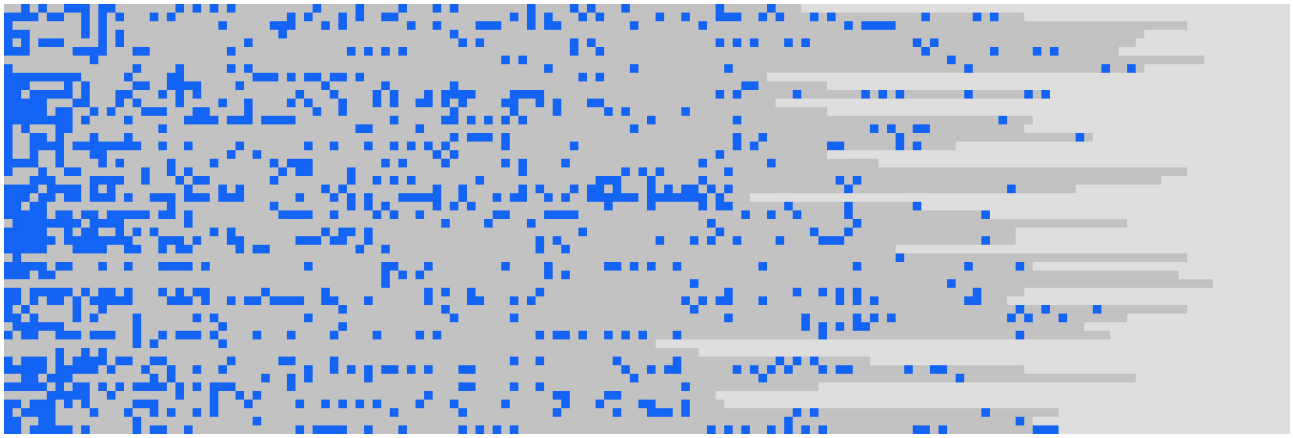

Fig. 5.8 4 Aggregated with Stemming and Stopword Removal turned on

This shift can also be seen when stopword removal and stemming is turned on in Fig 5.8.

In general it can be stated using the borda aggregation method increases relevance in relation to google search.

6 User Survey Analysis

Please see user Survey[20] in the Appendix B. I think the most interesting finding from the survey is that nearly everyone in it preferred the 4 search results presented side by side.

7 Problems Encountered

Accessing 3 Search engines APIs prooved a difficult challenge. I used Querypath[21] to scrape www.ask.com however it did not always give back reliable results. In the end I went with Bing, Yahoo and Blekko as my source search engines. Data Scraping was not needed.

Trying to figure out the most suitable aggregation technique was also quite difficult. Initially I was going to use the vector space model[22] and calculate the cosine similarity between the query and the snippet, however this seemed an unsuitable method. This method is more suitable to document collections with large bodies of text. I went with the Borda method because of its ease of implementation plus because it takes into consideration the current ranking of the documents well.

Yahoo's YQL[23] suddenly stopped working which didn't help. In the end I registered for Yahoo Boss to get guaranteed good quality results.

How to organise the application was a difficult challenge. I split the program into 3 seperate entities. These were the end user search(www.dandysearch.com), evaluation module www.dandysearch.com/index_evalution.php. There is also the data presentation module http://www.dandysearch.com/charts/select_charts_category.php

Running the evaluation was also proved quite troublesome. The Yahoo Boss server kept timing out when I was running my evaluation for the 50 queries. In the end I ran the full 50 queries in test 1. I ran 2 25 query tests for Test2. 2 25 query tests for test 3. I also ran 2 25 query tests for test 4. The evaluation also ran a lot faster when it was server to server rather than localhost to server.

20 https://docs.google.com/support/bin/answer.py?answer=87809
21 http://querypath.org/
22 http://www.miislita.com/term-vector/term-vector-3.html
23 http://developer.yahoo.com/yql/

8 Conclusions and Future Work

I learned a lot about object oriented Programming, PHP, and about the the model-view-controller software architecture. I improved my knowledge about the Document Object Model[24] and CSS. I also gained knowledge about different information retrieval methods. I touched on data scraping and the use of Regex to a small extend. I also got a good handle on the Bing, Yahoo, Blekko and Google APIs. I think I now have an idea of how you might go about putting together a large scale program.

This project differs from my intended project in that I did not get to implement either query expansion or clustering. I also did not implement nearly enough error checking and exception Handling. Also the aggregation method looks like it could be a lot more efficient. It involves a lot of deep copying. The efficiency of this method could be improved a lot.

Also when different snippets have the same bordascore, the order of there preference is not clearly defined. This really should have been improved upon. Some of the divs dont quite match up. The hack “ ” used to pull down the outer divs looks pretty dodgy. That could have been improved upon. Paging should really have also been added to display the results the standard 10 at a time. Some JavaScript could have been used to create a more dynamic feel to it ie drop down menus etc.

I should have used the class Search_Details in the a property in the scoreboard class to help minimize code duplication.

In general though I think this was an interesting and challenging project. I think I learned quite a lot in a relatively short space of time.

9 References Not Already included

Introduction to Information Retrieval. Manning, Raghavan & Schutze, Cambridge University Press. 2008. PDF version available online:

http://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf

24 http://en.wikipedia.org/wiki/Document_Object_Model

10 Appendix

Appendix A Raw Data from Evaluations

Appendix B User Study