# Experiences on a Practical Course of Web Information Retrieval: Developing a Search Engine

Fidel Cacheda, Diego Fernández, Rafael López
Department of Information and Communication Technologies
University of A Coruña
Facultad de Informática, Campus de Elviña s/n
15071 A Coruña, Spain
*{fidel, dfernandezi, rlopezga}@udc.es*

**A Web Information Retrieval course is quite appealing to Computer Science students and is quite challenging from the teacher's perspective, due to the limited knowledge of IR and Web IR of the students. In this paper we present our experience teaching a mainly practical Web IR course in order to exploit the programming skills of the common Computer Science student. The student is required to develop its own search engine and his evaluation is based on the features of the search engine implemented.**

*Keywords: Information Retrieval, Web Information Retrieval, teaching, education, practical course, search engine development.*

## 1. INTRODUCTION

Web Information Retrieval (IR) is a subject that results quite appealing to Computer Science students due to its reflection in well-known real life systems, such as Google[1] or Yahoo![2], and their interest to understand how these systems work. At the same time, teaching Web IR to generic Computer Science students is challenging due to their lack of knowledge in IR (and specifically in Web IR), or even worse, due to their partial knowledge of some aspects of Web IR (e.g. PageRank).

Starting on the point that theory and practice are fundamental for the student, one interesting aspect of a Web IR course is the balance between theory and practice, specially when considering generic Computer Science students.

The theory is fundamental to understand the structure of a search engine and the main issues to be solved in each component. On the other side, the students can easily get lost in the specific details and lose the global picture of the system, and their lack of knowledge of an IR system can make them over-simplify some of the problems discussed.

The practice is important to clarify and reinforce the problems analysed from the theoretical point of view. Systems like Lemur [3], Lucene [4] or Terrier [5] are very useful tools to test and study typical IR algorithms, but require a previous learning and could give the student only a partial view of a search engine architecture.

In this article, we would like to present our experience teaching a Web IR course to generic Computer Science Master students. This course was mainly practical and the objective was that the students developed their own search engine. In fact, the course was evaluated taking into account the characteristics of the search engine developed by the student.

The main motivation to teach this course mainly with practice was to exploit the programming skills of the students. Although the students may not have any previous knowledge about the basic IR models and issues, they can easily develop simple programs. In fact, the main components of a search engine (crawler, indexer and searcher) are quite simple programs that, with the appropriate guidance, could be easily developed by the students.

The remaining of the paper is organised as follows. Next we present the academic details of the course described here. Section 3 presents the course structure and details its contents. In Section 4 we briefly present the students' results and finally we conclude.

## 2. COURSE ACADEMIC DETAILS

[1] http://www.google.com/

[2] http://www.yahoo.com/

The course described in this article is named “Internet Information Retrieval” and corresponds to a second year subject in a Master degree in Computer Science at the University of A Coruña (Spain). It is an optional subject with a workload of 4 ECTS (European Credit Transfer and Accumulation System), equivalent to 40 class hours (25 theoretical + 15 practical) throughout 5 weeks in the beginning of the second term.

The first edition of this course was in 2007 and counted with 8 students, all of them graduate students. In the same Master there is a first year optional subject named “Information Retrieval Models and Techniques”, taught by professor Álvaro Barreiro, that is focused mainly in the basic models and techniques for IR. Three of the students had attended this course.

The Master in Computer Science at the University of A Coruña was a leading experience in order to adapt its subjects to the EHEA (European Higher Education Area), with a special focus in the work developed by the student.

Therefore, the main objective of our course was to lead the student to a clear understanding of how a search engine works and we considered that the best option would be that the student developed his own search engine.

## 3. COURSE STRUCTURE AND CONTENT

In this section we present the global structure of the course and the programming tasks that lead to the development of a simple search engine by the student. The implementation of the search engine is divided into three parts (crawler, indexer and searcher), where each one is associated with one module of the course (modules 2, 3 and 4, respectively). The programming tasks are recommended to be done in Java, as it is the programming language more used by the students, although it is not mandatory.

The course is divided in five weeks. The first week is dedicated to the introductory module and then, the next three weeks a new module is studied and a new programming task is assigned. The final week is used to solve problems appeared in the different programming tasks, do some improvements and, in general, integrate the components.

### 3.1 Syllabus

The syllabus of the course is the following:

- Module 1: Introduction to the search techniques
  - ○ Introduction
  - ○ World Wide Web
  - ○ Web search engines
  - ○ Architecture of a Web search engine
- Module 2: Crawling techniques
  - ○ Crawling algorithms
  - ○ Robot exclusion standard
  - ○ Robot meta-tags
- Module 3: Indexing techniques
  - ○ IR models
  - ○ Web indexing
  - ○ Distributed indexes
- Module 4: Ranking algorithms
  - ○ Content-based ranking
  - ○ Link-based ranking

The main references used in this course are the well-known [1], [2] and [6].

### 3.2 Crawler

The first module is a general introduction to the IR and Web IR and its main goal is to present a global architecture of a search engine to the student. In the next module, we describe the operation of a typical crawler and the politeness rules that crawlers must follow.

The programming task for this module is to develop a simple crawler that must have the capacity to crawl a small web site. In detail, the characteristics of the crawler are the following:

- Crawl our department’s web site[3], composed of approximately 3,500 web pages.
- Implement a deep first algorithm.
- Store the web pages downloaded in a repository. The student must define the appropriate data structure to store the web pages in order to the later indexing (e.g. URL of the page, id, title).
- Use a single thread, to avoid the saturation of the web server.
- Implement the robot exclusion standard.
- It is recommended to avoid downloading non-HTML documents (e.g. jpg, gif, pdf, doc) to speed up the crawling process.

[3] http://www.tic.udc.es/

The main problem when developing a crawler is to use the appropriate data structures in order to obtain a good performance and facilitate the programming. This is the main reason to introduce to the students some Java classes (and tips) that would help them:

- To work with the HTTP protocol, the programming class HttpURLConnection is recommended as it provides methods to access the main fields of the HTTP header.
- The main data structures of a crawler are the list of URLs to visit and the list of URLs visited. The class LinkedHashSet is recommended as it provides a good performance and keeps the insertion ordering of the URLs.
- In order to implement a simple parser of a Web page (to extract new URLs), we recommend the StringTokenizer class or the basic parsing methods of the String class.

The resulted crawler should be able to download and store the web site assigned in a few minutes (approximately, 3 minutes) from the department laboratories. The student will need only 15MB of disk space to store all the web pages downloaded in his repository (the common disk limit for students would be 500MB).

### 3.3 Indexer

The third module introduces the basic IR models (boolean, probabilistic and vectorial), the special characteristics of the Web indexing (e.g. word definition, HTML structural information, hyperlinks) and the basic index structure (vocabulary and inverted file).

The programming task for this module is the development of a simple indexer that will create the index for all the web pages previously stored in the repository. In detail, the main features of the indexer are the following:

- Divide the index data structure in two parts: the list of documents indexed and the index itself.
- The list of documents will store the basic information of each document (e.g. URL, id, title).
- The index will store, for each term, the list of documents where it appears (vocabulary and inverted lists).
- The whole index data structure will be stored into disk to be later used by the searcher.
- It is recommended to prepare an initial version of the index following the boolean model, and then, an extended version following the vectorial model. The different information required by each model is emphasized to the student.

As in the previous case, some suggestions are provided to the student in order to develop a, more or less, simple and efficient index:

- To store the list of documents any list can be used, as the access is done basically by document identifier.
- To store the index, it is recommended a HashMap because it will provide a quick access to the terms.
- As the number and size of the documents indexed is reasonably small, the whole indexing process can be done easily in main memory.
- To store the index data structure into disk, the Java serialization is recommended, as the disk space is not the main issue.

The resulted indexer should be able to index all documents stored in the repository in a few minutes (approximately, 5 minutes). The index will require about 5MB of disk space to be stored.

### 3.4 Searcher

The forth and last module presents to the student the main ranking algorithms, divided in content-based and link-based (e.g. HITS and PageRank) ranking algorithms. Finally, to complete the search engine, the programming task associated with this module is the development of a searcher that will use the index created previously.

In detail, the characteristics of the searcher are the following:

- Initially, a simple command line interface can be provided, where the user will introduce the search terms.
- The searcher must return a list of results, and for each of them, will provide its title and URL.
- Depending on the model used in the index (boolean or vectorial), the ranking should be done accordingly.
- Optionally, a web interface could also be provided.

In this case, no new data structures are required so there is no need to provide extra information to the students.

## 4. STUDENTS’ RESULTS

Based on our (short) experience from the first edition of this course, we would describe briefly the results obtained by the students.

From the eight students registered for this course, seven were able to develop a working search engine and get a successful mark for the course. The remaining student decided to not start the programming assignment due to lack of time. From the search engines presented six were developed in Java while one in Python (in this case, some performance problems appeared in the indexer). In Table 1 we present a summary of the main characteristics of the search engines developed by the students.

**TABLE 1:** Features of the search engines developed by the students

| | Crawler | Indexer | Searcher |
|---|---|---|---|

| | | Boolean | Vectorial | Command line | Web interface |
|---|---|---|---|---|---|
| Students | 7 | 4 | 3 | 4 | 3 |

## 5. CONCLUSIONS

Although our university regularly runs surveys through the students for each course, there is no official information available at the moment that we could provide. So, the conclusions stated in this section are based on our own experience and the comments received from the students in the classes, by e-mail or in the forum of the course.

The main advantage of the practical format of this course was the motivation obtained from the students to develop their own search engine. The students can get easily a minimal version of a search engine working and then, they can incrementally add new improvements to their system. For example, the boolean model (quite straightforward to implement) produces naïve results and so, the students try to implement the vectorial model in order to improve their ranking. Moreover, the students can understand more clearly the main techniques behind an IR system (e.g. the vectorial model), because they have to implement them and not only learn about them.

On the other side, some problems were detected during this course:

- In general, the students were not able to cope with the programming tasks weekly due to the concentration of the course in only 5 weeks. Instead, the students worked on the search engine during the whole term and the forum of the course turned out to be a very useful tool to solve their common questions and doubts.
- For a general Computer Science student most of the IR concepts are new and requires a higher guidance than expected for the development of the search engine.

## 6. FUTURE WORKS

In general we consider this experience as very positive and next year we will carry on with the same model for our course with some improvements. For example, to improve the guidance of the students, we would like to introduce a one hour class of “live” programming, for each programming task. In these live programming classes, we (the teachers) will implement a very basic version of each subsystem (crawler, indexer and searcher) and solve all the questions that may arise during the class. The resulting code will be provided to the students as a basic starting point.

In the next editions we intend to use some questionnaires to get some feedback from the students about the different aspects of the course (programming tasks, lectures, etc.).

## REFERENCES

[1] R. Baeza-Yates, B. Ribeiro-Neto (2002) Modern Information Retrieval, Addison Wesley.
[2] W. Frakes, R. Baeza-Yates (1992) Information Retrieval: Data Structures and Algorithms, Prentice-Hall.
[3] The Lemur toolkit for language modeling and information retrieval. http://www.lemurproject.org/.
[4] Apache Lucene. http://lucene.apache.org/.
[5] I. Ounis, C. Lioma, C. Macdonald, V. Plachouras (2007) Research Directions in Terrier: a Search Engine for Advanced Retrieval on the Web, CEPIS Upgrade Journal 8(1), Next Generation Web Search.
[6] I.H. Witten, A. Moffat, T.C. Bell (1999) Managing Gigabytes, Morgan Kaufmann.