Web Information Retrieval and Filtering Course to Undergraduates Using Open Source Programming

Seikyung Jung ▪ Joseph Lawrance

This paper describes how to engage actively students in web information retrieval and filtering course using open source programming. To teach this course, I utilized hands-on lab projects from various open source projects including the Galago search engine. Projects included, but were not limited to, implementing information retrieval (IR) algorithms, collaborative filtering (CF) algorithms, web-based interfaces, and adding features into an open-source search engine. By practicing with real-world open source programming, students found that they better understood how to connect background knowledge to real-world applications in preparation for industry jobs.

1 INTRODUCTION

One of the main shortcomings of computer science courses is the lack of practice with real-world systems, and the lack of a connection between real-world applications and background knowledge (Pedroni et al., 2007). If students fail to notice how the material they learn in class relates to exciting applications outside the classroom, they are more likely to lose the motivation necessary to continue studying (Knox et al., 2006).

While Web information retrieval (IR) is often taught at the graduate level (Murthy et al, 2008; Chau et al, 2003), it is increasingly taught at the undergraduate level as a special topic course (McCown, 2010). Collaborative filtering (CF) is an example of a statistical technique that forms the basis of much current research in data mining and information filtering. CF is used in many practical recommender systems such as for movies (http://movielens.umn.edu), books (http://www.amazon.com), scientific literature (ResearchIndex, http://citeseer.ist.psu.edu), and other domains (Jung et al., 2004). Some courses relate to CF recommenders (Alt et al, 2006; Reategui et al, 2007) and others relate to data mining (Musicant, 2006; Russell et al, 2006; Rahal, 2008). A few instructors reported on experimentation with open source software in the classroom (Pedroni et al., 2007). The nature of this interdisciplinary field has attracted experts, researchers and interested audiences from different fields such as database systems, artificial intelligence, data mining, human-computer interaction, and intelligent interfaces.

I wanted to develop a course for undergraduate students in CS with a minimal amount of prerequisites so as to maximize the course's potential audience. I made introductory Java courses the only prerequisites. This allowed sophomores, juniors, and seniors to enroll. While searching for a textbook, I discovered a book just released specifically to an undergraduate CS audience (Croft et al, 2009). I adopted the textbook for the course since it contains a number of exercises at the end of each chapter that use the open source search engine Galago, written in Java. I hope my experiences will be helpful to other Computer Science departments that are looking to develop an undergraduate Web IR course.

2 PROJECTS

The following is a sub-list of the various projects I covered in the class: lab, half-semester, and homework projects designed to give students hands-on experience. The projects that I describe form the major contributions of this paper.

2.1 Lab Projects

Hands-on lab projects taught students how to use the concepts I covered in lectures. At Bridgewater State University, we do not have separate lab classes or teaching assistants. Therefore, I allocate roughly half of my classes to lab projects and I give step-by-step instructions for students to understand how to install, navigate and run a large code base so that I could spend more time with students needing assistance. Samples of several lab projects follow.

2.1.1 Usability experiment using different kinds of IR Systems

Students focus on understanding what factors are important in choosing their primary search system from an end-users' point of view. The purpose of this experiment was for students to understand end-users better so that students knew where to focus when they actually developed IR systems. To conduct the experiment, students follow a similar procedure described by Jung et al (2008). Paired students use three different kinds of systems: commercial web search engines (e.g., Google, Bing), Google scholar, and Bridgewater State University's library search system.

To make projects appropriate for our students, I usually modify some of the source code. The topics include stemmer algorithms, encoding and decoding algorithms, the CF (e.g., Nearest-Neighbor) algorithm, and the PageRank algorithm.

At the end of the experiment, we discuss their experiences and their level of learning to make IR systems attractive to end-users. We also discuss ways to make IR systems better from developers' perspectives. Students notice many factors (e.g., range of material types, familiarity with search system, access to full text, description of results, advanced filtering, spell checker) that influenced their discussion about which IR systems are better not only for ranking (effectiveness), but also for speed (efficiency), and ease of use.

2.1.2 Install Galago Search Engine in Linux Virtual Box

In this lab, I require students to install Oracle VirtualBox[1] with Linux (Ubuntu) on their individual laptops and install the Galago search engine. To run Galago properly, students need to install additional software and modify existing source code. This experience turned out better than my original plan for students to share the department web server. Students appreciated learning how to set up Linux and install necessary software as the system administrator (root) through step-by-step instructions. For most students, this was their first experience with Linux, not only as a user but also as a sys-admin. Although using the department web server would have been easier, students learned more by administering their own Linux system in VirtualBox.

2.1.3 Experiments on given IR components

After learning various concepts of IR systems, students had a chance to read and browse source code. For some components, we use Galago source code; for others, we use open source code from another book (Segarn, 2007). To make projects appropriate for our students, I usually modify some of the source code. The topics include stemmer algorithms, encoding and decoding algorithms, the CF (e.g., Nearest-Neighbor) algorithm, and the PageRank algorithm.

In most of these projects, students execute the given source code on their Linux machine with my step-by-step instructions. They calculated output manually based on what they learn in class. Then, they compare their hand-calculated output with program-generated output. For these labs, I expect students to understand how each algorithm works and affects search results. Although students did not implement these algorithms, students did appreciate learning how algorithms work during lab time. Students also appreciate the importance of evaluating each algorithm by hand for testing and debugging purposes. A brief summary of some of the projects follows:

- In stemmer projects, students compare the output of Krovetz and Porter stemmer algorithms from several webpages. Students debate under which circumstances each algorithm works best after reviewing source code and running experiments. For this lab, I expect students to understand the pros and cons of each algorithm after the experiment rather than learning just the theory through lecture.
- In encoding and decoding projects, students modify a simplified codec so that it would work as a VByte codec. Since search engines index large collections of webpages, students must understand how to compress data using various encoding and decoding schemes. In this lab, pairs of students freely discussed how to modify code. They prove that their code works properly by comparing hand-calculated output with program output.

[1] http://www.virtualbox.org/

- Movie recommender with CF algorithm: In this lab, students use the Nearest Neighbor (NN) algorithm with a publicly available movie dataset[2]. Students run the source code and then compare the program output to computing the results by hand. Students learn how to calculate NN and predict scores. Students also build a del.icio.us[3] link recommender. Students with del.icio.us accounts are able to get neighbor and link recommendations from their own bookmarks.
- Link analysis algorithms like PageRank (Brin and Page, 1998) are designed to provide more reliable ratings of webpages. Google initially used PageRank to weigh relative importance of each webpage. In PageRank projects, students calculate PageRank manually for a small set of webpages based on what they learn in class. Then, they compare their manual calculations to their PageRank program output.

2.2 Half-semester Group Project

One component of the course is a half-semester-long group project. Each group is responsible for defining its own project. I recommend a group size of two, but groups of three students also are acceptable for projects that are more ambitious. For the backend, students use one of the following languages: Java, C++, or Python. For the frontend, students could use anything. For the group project, students submitted a pre-proposal, first and second progress reports, and a final presentation.

The followings are examples of past students' projects.

- *Implementing a Bridgewater Restaurant Recommender*
  Students built a website that provides CF recommendations for Bridgewater restaurants. Students came up with a list of all Bridgewater restaurants, and found a way to collect ratings for the restaurants so that students could test their system.
- *Image search integrated into Galago*
  Students added crawler features for image search into Galago and modified existing code so that users can choose either web or image search.
- *Implementing and evaluating ranking algorithms (e.g., TF/IDF, BM25) in Galago*
  Students found that it was difficult to evaluate different kinds of ranking algorithms in Galago. One group modified existing Galago code to ease evaluation. They implemented different kinds of ranking algorithms and compared performance.

2.3 Homework projects

I prohibit group work in homework projects since students work as groups in lab projects and in the half-semester group project. In the first assignment, students describe how the Galago search engine works. By navigating source code, students match core IR components to source code files (.java). This requires students to use Eclipse features (e.g., Find References and Go To Declaration) to navigate approximately 38,000 lines of Galago source code. Most homework projects are from exercises at the end of each chapter. The topics include implementing (and integrating into Galago) a crawler, a tokenizer, and BM25 ranking algorithm.

[2] http://movielens.umn.edu
[3] http://www.delicious.com/

3 DISCUSSION AND STUDENT FEEDBACK

From an instructor's point of view, creating lab projects based on open source software takes much more time than preparing lectures or homework projects. Gaining familiarity with open source code and tailoring the code to a level appropriate for student lab projects is as rewarding as it is time-consuming. Existing open source code can be overwhelming for students, unusable, or unavailable, meaning that for some projects, I had to write some code from which students could start their work. Furthermore, finding open source code suited to special topics (e.g., collaborative filtering) to complement Galago code required substantial effort.

I prohibit group work in homework projects since students work as groups in lab projects and in the half-semester group project. In the first assignment, students describe how the Galago search engine works.

Students appreciated their open source code experience as demonstrated by these testimonials.

> "I like it and learned more. If there was commenting I could have learned even more and been able to work faster"

> "It was initially overwhelming. But once I understood the code, it became attaching new blank screens to the project and filling those screens. I did learn more because I had to understand the code of someone else before I could start coding"

> "I learned more because I felt like I was doing something useful"

From a learning perspective, the most memorable part of the course for most of the students was the hands-on lab projects. For many of them, this is the only course where they had the oppor-

tunity to learn hands-on programming and dealing with a Linux system in class. Students commented that it was extremely helpful to have a lab class right after lecture class, because they gained experience with real-world code, and learned from comparing hand-calculated output to program output.

> "This was my first CS course with lab activities and I must say that it has also been the most helpful CS course I've ever been in. They were great!"

> "They were helpful to me because I learned better when I do hands on activities, rather than sitting through a lecture"

> "They helped me understand the algorithms greatly. Without actually doing the algorithms, I would have no idea what all the random letters meant"

Most students appreciate having half-semester long projects because they could freely choose the topic and gain open source programming experience. One student commented that he had a job interview and the interviewer gave him a Galago-size program and asked him to explain how it works within twenty-four hours. He mentioned that the code did not have any documentation or comments. The Galago experience helped him to navigate the code better (installation, navigation through Eclipse).

> "This gave me experience with working on a long term development project as well as let me be creative. I really liked this idea"

> "It was nice since we get to work on something related to class and related to our lives"

> "It was helpful to learn how to work on a project containing large amounts of code"

Even though I did not ask students to build the search engine from the ground-up, some students had a hard time implementing some of the projects. These students were overwhelmed by the size of Galago, even though similar projects such as Lucene and Nutch dwarf Galago's size. (Lucene and Nutch[4] have approximately 177,000 and 61,000 lines of code, respectively). By the end of the semester, students felt better about navigating and modifying Galago. One of the reasons that I chose to give most of the code and asked to modify existing source code is because most of time, when students get hired at a company, the first thing they do is understand existing code, fix bugs, and modify code to add features.

> "Half-long-semester project was very stressful. The project was overwhelming"

[4] Nutch (http://nutch.apache.org/) is an open source search engine using Lucene (http://lucene.apache.org/) API.

Finally, I should mention that I also assigned two midterm exams and one final exam. The course that I have described above has been quite successful. Students indicated in evaluations that they really enjoyed the opportunity to combine "real-world" problems with computer science theory and algorithms. I have found that this course helps considerably in preparing students to work in industry jobs. All of my course materials is available upon request (sjung@bridgew.edu). **Ir**

References

[1] C. Alt, O. Astrachan, J. Forbes, R. Lucic, and S. Rodger. Social Networks Generate Interest in Computer Science. *Proceedings of the 37th ACM SIGCSE technical symposium on Computer Science Education* (2006), p438-442

[2] S. Brin and L. Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. Proceedings of the 7th international conference on World Wide Web (1998), p107-117

[3]]M.Chau, Z.Huang, and H.chen. Teaching key topics in computer science and information systems through a web search engine project. *ACM Journal of Educational Resources in Computing (JERIC)*, 3(3):2, 2003

[4] B. Croft, D. Metzer, and T. Strohman. Search Engines: Information Retrieval in Practice. Addison Wesley, 1st edition, February 2009.

[5]]S. Jung, J. Webster, M. Mellinger, J. Frumkin and J. Herlocker, LibraryFind: System Design and Usability Testing of Academic Metasearch System. *Journal of the American Society for Information Science and Technology (JASIST)*, 59(3): 1-15, 2008, John Wiley & Sons Inc, NJ.

[6] S. Jung, K. Harris, J. Webster and J. Herlocker, SERF: Integrating Human Recommendations with Search. Proceddings of the 13th conference on Information and Knowledge anagement (CIKM), p571-580, 2004, New York, NY: ACM Press.

[7] D. Knox, P. DePasquale, S. Pulimood. A Model for Summer Undergraduate Research Experiences in Emerging Technologies. *Proceedings of the 37th ACM SIGCSE technical symposium on Computer Science Education* (2006), p214-218

[8] F. McCown. Teaching Web Information Retrieval to Undergraduates. *Proceedings of the 41st ACM SIGCSE technical symposium on Computer Science Education* (2010), p87-91

[9] U.Murthy, R. Torres, F. Edward, V. Logambigai, S. Yang, and M. Goncalves. From Concepts to implementation and Visualization: Tools from a Team-Based Approach to IR. *Proceedings of the 31st annual international ACM SIGIR conference*, (2008), p889.

[10] D. Musicant. A Data Mining Course for Computer Science: Primary Sources and Implementations. *Proceedings of the 37th ACM SIGCSE technical symposium on Computer Science Education* (2006), p538-542

[11] M. Pedroni, T. Bay, M. Oriol, and A. Pedroni. Open Source Projects in Programming Courses. *Proceedings of the 38th ACM SIGCSE technical symposium on Computer Science Education* (2007), p454-458

[12] I. Rahal. Undergraduate Research Experiences in Data Mining. *Proceedings of the 39th ACM SIGCSE technical symposium on Computer Science Education* (2008), p461-465

[13] E. Reategui, E. Boff, and J. Campbell. Using Virtual Characters in Personalized Recommendations. *Proceedings of the 38th ACM SIGCSE technical symposium on Computer Science Education* (2007), p180-184

[14] I. Russell, Z. Markov, T. Neller. Teaching AI through Machine Learning Projects. Proceedings of the 11th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE, 2006), p323

[15] T.Segaran, Programming Collective Intelligence. O'Reilly Media, August 2007.

SEIKYUNG JUNG
Mathematics and Computer Science Department
Bridgewater State University
Bridgewater, Massachusetts 02325 USA
sjung@bridgew.edu

JOSEPH LAWRANCE
Department of Computer Science and Networking
Wentworth Institute of Technology
Boston, Massachusetts 02115 USA
lawrancej@wit.edu

Categories and Subject Descriptors: H.3.3 [Information Systems]: Information Storage and Retrieval; K.3.2 [Computers and Education]: Computer and Information Science Education - Computer science education
General terms: Human Factors, Experimentation
Keywords: Information retrieval, computer science education, open source programming

DOI: 10.1145/2003616.2003634