A Course-Long Information Retrieval Project

David Kauchak
Pomona College
185 East Sixth St.
Claremont, California 91711

Abstract

In this paper, we describe the outline for a course-long information retrieval (IR) project. The project guides the students in constructing a working IR system from the ground up. The first half of the project is structured and closely follows common foundational IR concepts. During this portion of the project, a bare-bones IR system is constructed. For the last half of the project, students (in groups) implement research-driven extensions to the basic system with the additional constraint that their project must integrate with the base system. By the end, the students have worked on a large software project ($\sim$40 classes with thousands of lines of code) in a group setting as well as been introduced to the research process. This project plan has been successfully used in an undergraduate course; resources including starter code, solutions, and an example IR system with project write-ups are available.

Introduction

Information retrieval (IR) is the study of searching and processing document collections, where a document is a web page, news article, blog, tweet or even textual information in a database (Manning, Raghavan, and Schutze 2008). IR courses have been increasingly appearing in the curriculums of computer science departments as their impact in society has grown and more resources have become available for instructors. IR courses allow the instructor to explore an application area that combines components from a range of fields including artificial intelligence, natural language processing, machine learning and linguistics.

In this paper, we introduce a course-long project that can be used in an undergraduate or graduate IR course. The first half of the project is highly-structured and consists of four individual assignments. The assignments are cumulative and build upon the code and concepts from the previous assignments. The first three assignments construct a basic IR system which can perform both boolean queries and free text queries. The fourth assignment explores different evaluation metrics, the evaluation of the built system and how system variations/parameters affect performance.

While the latter half of the project could also be setup in a similar fashion, with structured assignments, we have found that from institution to institution, IR courses tend to emphasize different concepts. For the latter half of the course, the project is more open ended and allows the instructor to emphasize concepts that are relevant for that particular course. Students in groups design, implement and evaluate course specific additions to the basic system. The student projects include three constraints: 1) it must be related to something discussed in class 2) it must extend and integrate with the code developed in the first half of the project 3) it must be evaluatable and there must be a plan for how to measure the performance of the addition. These three requirements reinforce our course goals, which we discuss below. The students then plan out and implement their additions, write a short paper and give a presentation describing their project and results.

What we layout through the rest of the paper is one approach for a course-long IR class project which emphasizes the concepts and skills we feel are important. Depending on the goals of a particular course, the overall project can easily be adapted to meet the particular goals of the course. As much as possible, we try to be explicit about why we chose our particular approach and how it addresses our underlying goals.

In developing this project, we had three main goals in mind. First, we wanted a project that would reinforce and apply the IR concepts being taught in the classroom. The initial assignments closely follow common, foundational concepts taught in many IR courses and can be modified to emphasize different concepts if desired.

Second, we wanted to introduce students to some of the experiences encountered when developing and working on a large software project. Because the assignments are cumulative, by the end of the first four assignments, the students have developed a system that contains over 20 classes with around 2400 lines of code (by the conclusion of the project, the code base for a class of 13 students had $\sim$40 classes and over 6000 lines of code). Much of this code is written by the students, but some is provided for them, modeling a more realistic setting where they must understand and extend existing code that they did not write themselves. For the last half of the project, we model the project after a larger software setting where the students work in small groups on small

Figure 1: Screen capture of the output from a system developed over a semester based on the project described in this paper.

projects, but must coordinate globally to create a final, integrated system. During this period, we force the students to think about the challenges of coordination and integration and also explore tools like code repositories.

Third, we wanted to introduce students to the research process. We encourage students to quantitatively evaluate their project and then present their results both in writing and orally. These tasks force the students to think beyond just the implementation details.

By the end of the course, the students develop a functional IR system with many of the features of modern IR systems. Figure 1 shows a screen capture of sample results of a system designed by students. The system had very efficient retrieval and supported snippets, image retrieval and the PageRank document weighting scheme.

Our goal in this paper is not to describe concepts that should be taught in an IR course. (McCown 2010) provides a good analysis of material commonly taught in an IR course and (Mizzaro 2007) a discussion about differing approaches to teaching an IR course. Instead, we provide a flexible project framework, including code and descriptions, that can accompany a variety of IR syllabi. We describe our experiences in teaching with these resources and suggest how they can be adapted to meet other course content and design goals.

The paper is outlined as follows: we first give a brief overview of common content in an IR course and describe how the project integrates with this curriculum. We then describe the initial set of assignments followed by a description of the final project. We have used this project in a course and detail our experiences and describe what resources are available for those interested in following a similar approach. Finally, we conclude with final comments and future extensions we plan to deploy.

IR Course Overview

To help better understand how the project can complement the material covered in an IR course, we present a rough outline for the content of a common IR course here:

- **Text processing** - One of the key challenges with many applications is preprocessing the data. For text, this involves a variety of tasks including tokenization, casing, normalization, stemming, lemmatizing, stop word removal, file format detection and parsing, etc.
- **IR systems** - IR systems come in a variety of types from boolean query systems to full text querying with a range of features and capabilities including positional search, phrasal search, wildcards and zone search to name a few.
- **Index construction** - Because of the size of data sets used for IR (e.g. the web) and the requirement for near instantaneous retrieval, one of the core components of an IR course is discussing how to efficiently build an index that allows for quick retrieval of the desired results while also accommodating a wide range of functional features.
- **Evaluation** - While often not apparent to many IR system users, a crucial part of building an IR system for either research or industry is how to evaluate the performance of the systems on a broad range of metrics including speed, size, user happiness, ease of use and quality of the results.
- **IR and the web** - Although IR systems cover a range of different applications areas, one of the most popular applications is searching the web. Web search has a number of uniques challenges that must be overcome because of its size, variability and dynamic nature.
- **Beyond basic IR** - There are a variety of additions to an IR system as well as applications that are related to IR systems that can often be covered in an IR course. Additional concepts include searching alternate media (images, audio and video), query expansion, relevance feedback, snippet generation and cross-lingual IR. IR also leverages many applications in natural language processing and machine learning such as classification, clustering and modeling.

Most IR courses will cover the first five bulleted topics in varying detail (for example, these roughly correspond to the first 8 chapters in (Manning, Raghavan, and Schutze 2008), a common IR textbook). For more detail on these topics there are many textbooks available. We suggest the following two as a good starting point (Manning, Raghavan, and Schutze 2008; Grossman and Frieder 2004). The book web

page for the former contains many resources on other IR books, current courses, problem sets, etc.

Depending on the emphasis and length of the course, a variety of topics can then be covered that examine extensions to a basic IR system. For the course-long project, this portion of the course plays an important roll in introducing students to the types of research that occur in the field of IR as well as initiating the brainstorming process for candidate projects/extensions to the basic system. One of the benefits of the project we outline in this paper is that the direction that the students (and the instructor) may take for the latter part of the course are flexible.

Initial Assignments

In this section, we outline the initial assignments that explore the steps involved in building the basic IR system. These projects mostly involve coding, but can also include additional questions based about the students' experiences. The assignments are incremental and build upon the code base from the previous assignments[1]. In our experience, two weeks was a reasonable amount of time for each assignment. For many of the assignments, the difficulty and time required can be varied by giving the students more or less initial starting code or by adjusting the different features to be implemented. By the end of the initial assignments, the students will have implemented a working IR system and have the tools available to evaluate the performance of the system and system variations. In the sections that follow, we describe the initial assignments that build and evaluate the base IR system.

Assignment 1: Text processing

The first step for an IR system (and for most natural language processing systems) is to to read in the text, break the text into tokens and then apply token normalization techniques:

- **Reading text:** For large data sets, an important step is how to iterate over the different documents in the data set. This process helps students understand I/O challenges and also introduces them to the challenges of handling formatting constraints. In addition, this portion of the assignment reinforces interfaces and good coding practices since we want to develop our system for different underlying data sets.
- **Tokenization:** Once the documents are processed, each document is represented as a long string of text. Given this text, the next step is to break the text into tokens. The tokenization step involves proper handling of quotes, numbers, abbreviations and other punctuation. This portion of the project also introduces students to regular expressions and their uses.
- **Token normalization:** During the normalization step, tokens that represent the same concept are normalized to a common token. Normalization steps including lowercasing, stemming, number folding and stop word removal, all of which are common text normalization techniques that show up in many other applications.

[1]During the course, we provide to the students solutions to the previous assignments as we move forward, since they are incremental.

Modifier	Vocab size
simple tokenization	198,233
improved tokenization	114,295
number folding	108,027
lowercasing	95,563
stemming	91,272
stop list	114,076
num. folding, lowercasing, stop	89,080
all	68,302

Table 1: Example vocabulary sizes for different preprocessing settings on a sample data set. All results are reported using the improved tokenization except the first entry.

All of the steps above are parameterized and different variations of tokenization and normalization can be compared. After the coding, the students examine the impact of the different text processing techniques on the size of the vocabulary for a data set. Even for moderately sized data sets, the impact of the different techniques is often dramatic. Table 1 shows example results comparing the different techniques on a 15K document data set. By the end of this first assignment, the code-base already consists of 8 different classes that the students utilize.

Assignment 2: Boolean IR system

The next assignment is to build a boolean retrieval system where the queries are a combination of words and operators, including `AND`, `OR` and `NOT`. As mentioned, the assignment builds directly on the code developed in the first assignment. In a boolean retrieval system, there is no ranking of the documents, a document either matches the query or it does not. This is accomplished by creating a boolean index which consists of a mapping from tokens to postings lists, where the postings list represents all of the documents that token occurred in. The postings lists are implemented using a linked list, allowing the students to see a practical use of their data structures coursework. To support the different boolean operators, methods that *intersect*, *union* and *not* postings lists are implemented. The algorithms to support these operations are a common topic when discussing index construction and further test the student's ability to manipulate linked lists.

Once the index and posting lists are created, the query must be parsed and executed on the index to obtain the final list of matching documents. For simplicity, the basic assignment only requires handling of non-complex queries, that is, queries that do not contain parentheses nor are the students asked to do attempt to optimize the query processing ordering for performance. These additional features would be possible alterations to the basic assignment to make it more challenging, for example for a graduate course, or as an additional assignment.

Assignment 3: Ranked IR system

Boolean systems are common in some domains, however, most modern IR systems are based on a vector-based approach where documents are represented as word count vectors. The query is then also represented by a word count vector and documents are ranked by a measure of similarity between the query vector and the document vector (often some variant of the cosine similarity measure). Systems rarely use the raw word counts to represent a document and a variety of normalization and weighting schemes are also examined to improve the quality of the results.

To accomplish this, students first modify the existing index and posting lists to include additional word frequency and word weighting information. As with the tokenization, there are many variations on how the document vectors are constructed: the word frequencies can be altered, word weightings applied and different vector length normalization techniques employed. We suggest exploring at least two options in each of these categories so that students see the impact of these different factors. Again, this assignment complements common algorithms discussed in an IR class and allows the students to understand why certain algorithmic decisions are made in the methods discussed in the classroom.

In adding the vector-based approach, the students are required to continue to support boolean queries, which can be done on the same, though slightly modified, index. The students can then compare the performance differences when issuing a boolean query where occurrence frequency and word importance are not taken into account, versus the vector space approach which do include these additional features.

By the end of this assignment, the students have constructed an IR system that has many of the key components of most commercial systems. There are still issues that would need to be resolved for scalability, but the size of data set that can be processed is sizeable and this can also help students understand the need for parallelized indexing approaches.

Assignment 4: Evaluation

For the final structured assignment, the students explore the evaluation of the system. In the IR literature, there are a number of different evaluation measures commonly used including precision, recall, R-precision, mean average precision (MAP) and normalized discounted cumulative gain (NDCG) to name a few. Each of these evaluation measures highlight different characteristics about the system's performance and can be useful in better understanding the behavior of the system.

For the system described above, there are many different parameterized options that can be selected between when deploying a final system, including text preprocessing variations, weighting schemes and index normalization techniques. After implementing some of the above evaluation metrics, the students then experiment with different parameter settings to better understand their impact on the performance of the system. Table 2 shows example results from the system when varying different index normalization techniques. The Cranfield (Cleverdon 1967) data set is used because the data is freely available, however, much larger data sets are also available through the LDC[2]. The results show a marked difference in the performance of the different techniques and also highlight the discriminative ability of the different evaluation measures. Similar results (though less dramatic) can also be seen when examining the impact of text preprocessing techniques on system performance.

Final Project

The initial assignments develop a bare-bones IR system. After completion of the assignments, the students have spent a substantial amount of time developing and familiarizing themselves with the code base. For the final project, the students, in groups, propose, implement and report on, research-driven additions to the basic system.

We have three main goals for this final project. First, we want the students to implement and apply some of the additional concepts discussed in the later part of the course. Because of the variety of material available for an IR course, it is generally infeasible to construct assignments/projects on all of this material. Instead, the students can study in detail, one aspect of IR that they find interesting.

Second, we found this was a good opportunity to explore the concepts and challenges of developing a large software project. Each groups' project is required to integrate with the code base developed in the initial assignments. In the project proposal, the students plan out what changes are required to this code and how they will integrate with the existing code. The students work in small groups, requiring both coordinating within groups and coordination between groups. Depending on the goals of the course, the instructor can play the role of facilitating inter-group planning and coordination or this can be an additional role for the students to take on. SVN (or other version control software) is used to keep track of the state of the code. For large class sizes, it may be easier to split the class into multiple larger groups, where each group is developing a stand-alone IR system.

Third, we want the students to explore the research process in the IR field. All of the projects are required to be evaluatable. For many projects, this is straightforward, for example, a team implementing index compression can measure the decrease in the size of the index and a team implementing query optimization can measure the decrease in query time. However, some projects, such as GUI development, can be more challenging. At the end of the project, the students submit a research-oriented write-up and give an oral presentation discussing their approach and results.

Below are some suggestions for possible projects:

- GUI development
- Query optimization
- Query suggestion/rewriting
- Index compression
- Faster/approximate ranked retrieval

[2] http://www.ldc.upenn.edu/

Count normalization	Term weighting	Length normalization	Precision at 20	Recall at 20	R-Precision	Map
none	none	none	0.006	0.052	0.002	0.015
log	none	none	0.029	0.224	0.060	0.076
none	IDF	none	0.046	0.390	0.109	0.158
none	none	cosine	0.037	0.308	0.100	0.131
log	IDF	none	0.055	0.455	0.132	0.189
none	IDF	cosine	0.065	0.539	0.148	0.218
log	IDF	cosine	0.068	0.547	0.166	0.229

Table 2: Sample evaluation results when varying index normalization techniques. Results are based on the Cranfield data set.

- Relevance feedback
- Result clustering
- Result classification
- Web crawling
- Positional index/phrasal queries
- Wildcard/regex support
- Document segmentation
- Snippet generation
- Parallelized indexing
- Parallelized index querying
- Multimedia search (image, audio, video)
- Document importance (e.g. PageRank)
- Advertising

Experiences

We used this project plan to teach a course with 13 students during an undergraduate, semester-long course on information retrieval at Pomona College. The initial assignments required roughly two weeks each, which took us halfway through the course. The assignments were implemented in Java, which was chosen because the end result of the course project is sizeable and the object-oriented and structured nature of the language was very important for maintaining good software development practices. Many students commented after the completion of the course that they found the project to be a very useful exercise to reinforce their previous courses in Java.

For the last half of the course, we had six different project groups. One group worked on the GUI, two groups worked on snippet generation, two groups worked on image search (one based on text search and the second on image-based search) and the last group implemented and integrated PageRank. All of the groups except one of the image groups were able to integrate their code with the original code base. The resulting final project had approximately 40 classes with over 6000 lines of code, an increase of over 3500 lines of code from the base system.

We encouraged students to do what they found interesting. In some cases, this meant that we had multiple groups tackling the same problem. While this did not result in more features for the final system, it did allow the groups to share evaluation data sets and to compare results between differing implementations. For example, the two teams that did snippet generation collaborated to construct a data set for evaluation. In both their papers and their presentations, they were able to compare and contrast their results with those of the other team.

The students found assignment 1, the text processing assignment, to be the least exciting, but did gain an appreciation for the difficulty in dealing with text data. They found assignment 3, the ranked IR assignment, to be the most challenging assignment conceptually, mostly because of the techniques required to normalize counts over an index efficiently. The easiest assignment was the fourth assignment, evaluation, though this assignment gave the students the most opportunity to understand what the system was doing well and how different component affected performance.

To coordinate the final project, we spent one class where each of the groups discussed their suggested additions and changes to the system and we were able to coordinate a majority of the interactions during this one class. Surprisingly, there were very few integration issues, though this may have been a factor of class size. The most challenging project to integrate and deploy was the GUI project, since experimenting required it to be deployed on an actual web server. In general, the students found the implementation to be straightforward, but were challenged in attempting to properly evaluate the performance of their project.

Resources Available

There are a number of resources available for anyone interested in implementing this project in a course. For each of the assignments, a step by step description is available describing (for the students) what is required and how it should be accomplished. Along with the assignment descriptions there is starter code in Java for each assignment as well as code for the solution. For the final project, an example project description is available. Finally, live demos are available for other courses that have followed this project structure as well as white papers written by the students describing their contributions to create the system. All of this is publicly available[3]. Since the starter code for the one assignment is often the solution to the previous assignment, we have not published the starters or the solutions online, how-

[3] http://www.cs.pomona.edu/∼dkauchak/ir_project/

ever, the authors can be contacted to gain access to these resources.

Conclusions

We have described a course-long IR project that allows students to apply their classroom knowledge to build a working IR system. In addition, we have suggested one approach for allowing students to extend this basic system based on research interests within the field. These additions not only allow the students to explore a research area, but also expose them to some of the challenges in developing a large software project. Many students have commented that the experience was valuable and found it a useful project to discuss in job interviews because of the size, scope and research-driven design of the project.

We have described our goals and motivation for our suggested path, however, many of the assignments and resources can also be used individually. In addition, the open endedness of the final project allows for a great deal of flexibility in designing an IR course.

There are a number of additions to the project that could make it more useful. For many of the assignments, there are additional extensions that could be used as variations or additional assignments. Many of the suggested final project topics could also be straightforwardly integrated to create an additional assignment. Another interesting question would be to compare the differences in implementation and performance between open-source IR systems such as LuceneLucene[4] or Nutch[5]. Finally, one of the challenges with the current project specification is that visualizing returned results is problematic. We are in the process of developing a suitable UI and hope to make it available soon.

So far, we have only used this framework in one course and our experiences described above are qualitative. We hope to gather more information about the effectiveness of this type of approach by examining experiences at other institutions as well as taking a more quantitative approach to evaluating the project's success.

References

Cleverdon, C. W. 1967. The cranfield tests on index language devices. *Aslib Proceedings*.

Grossman, D., and Frieder, O. 2004. *Information Retrieval: Algorithms and Heuristics*. Springer.

Manning, C.; Raghavan, P.; and Schutze, H. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

McCown, F. 2010. Teaching web information retrieval to undergraduates. *SIGCSE*.

Mizzaro, S. 2007. Teaching of web information retrieval: Web first or ir first? *TLIR*.

[4] http://lucene.apache.org/

[5] http://lucene.apache.org/nutch/