A Technical Approach to Information Retrieval Pedagogy

Rafael López-García, Fidel Cacheda

Department of Information and Communication Technologies. University of A Coruña. Facultad de Informática, Campus de Elviña s/n. 15071 A Coruña, Spain.

Abstract

Like in other subjects belonging to the Computing Science curricula, learning in Information Retrieval must be significant. In order to understand the main concepts and procedures of this discipline, students and inexperienced researchers must acquire some practical skills which often are demonstrated by means of the transformation of a specification into a software product. In order to reach those aims, this article presents a technical approach to Information Retrieval teaching that focuses on the links between theoretical concepts and programming code. It also presents IR-Components, a framework that could facilitate this purpose.

Introduction

Computing Science (CS) is a young discipline whose curricula have to be constantly renewed and tend to be divided into different specializations [1]. Information Retrieval (IR) is one of them and syllabi in IR are usually disparate too. Duration can oscillate from a few sessions to months and students with no previous knowledge of the subject or even of programming can fulfil the admission criteria. Actually, as IR is a multi-

disciplinary science, approaches can be really heterogeneous, from holistic [2] and philosophical [3] to detailed and technical [4].

Fortunately, experts in technical IR identify two components of IR systems: the indexer and the search engine. A third one, the crawler, is usually added in Web environments. There are lots of common and interesting concepts and techniques related to those components [5, 6, 7] (e.g.: MIME [8] types, robots exclusion standard [9], stop words, stemming [10], weighting models [11] and so on). Teachers have to decide how many and which concepts are going to be part of their syllabi.

Whatever the characteristics of the subject, there is a significant need for methodologies and teaching materials in order to improve students' understanding. Most of the classic systems, like Apache Lucene [12] and Terrier [13], are too complex to be understood or changed by inexperienced students. There are some simpler and pluggable initiatives, like IR-Base [14], but the number of systems fulfilling these features is not high.

The first goal of this article is to present some methods which help technical teachers to solve some of their problems and to facilitate several tasks like, for example, presenting the main problems of IR to their students, guiding them to the solution of these problems, establishing their coursework and evaluating their theoretical and technical comprehension.

Another aim of this article is to present the IR-Components system as a candidate to help teachers to solve the proposed problems and to fulfil the aforementioned needs. The system is specially oriented to this purpose, since it is simple and its design pays special attention to modularity.

Related Work

Among the multiple disciplines involved in IR, Library and Information Science (LIS) has become one of the most important since 1951, when Mooers introduced the term in that context [15]. Nevertheless, IR makes use of computer-related technology, and that was the reason for IEEE & ACM to include it in the CS curricula [1]. Some authors like Zhu and Tang [16] presented their own proposals for degree and post-degree levels too.

Fernández-Luna et al., who have recently published a literature review of pedagogical methods for teaching and learning IR [17], have identified three main educational goals about IR in the CS curricula and literature. The first one, understanding fundamental aspects of IR, has already been developed by some authors, like Henrich and Morgenroth [18] and Efthimiadis and Hendry [19, 20]. Other authors go into higher detail and teach advanced techniques, like Herrera-Viedma et al. in fuzzy systems

[21] and Goharian et al. [22] in data mining, but those concepts are out of the scope of this article.

The second one, training in search strategies, is more typical of the LIS curricula. In this kind of syllabi, teachers usually show students how to choose among different search engines and how to exploit their features, but they never force them to program their own solutions and hardly ever teach technical details (although there are exceptions like Johnson [23]).

Besides, Airio et al. presented a tool for measuring the features of IR systems [24] in order to improve students' search skills.

Finally, the last aim Fernández-Luna et al. found, to acquire skills to develop new IR methods using software modules, has also been researched by several authors. Efthimiadis and Freier proposed an approach in which students do not have to program [25]. Jinguji et al. designed a customizable answering system [26], but it is not exactly a document retriever and its complexity makes it oriented to advanced courses. De Campos et al. presented an object-oriented framework for the research and teaching of IR [27], but this system is oriented to Probabilistic Graphical Models (PGM) and structured document formats like XML. Chau et al. submitted another initiative [28], but their proposal for the coursework is to give two complete applications to their students, AI Spider and AI Indexer, and make them develop the search engine starting from scratch to get a medium/large scale project. IR-Components, on the contrary, forces students to understand each of the applications by means of completing a skeleton, programming only the most illustrative parts and focusing on the essential concepts and techniques of IR chosen by the teacher.

The main alternatives to IR-Components seem to be IR Framework [29], described in 1994 by Wade and Braeckevelt, and IR-Base, presented in 2007 by Calado et al. [14]. Unfortunately, in the case of IR-Base, the authors do not provide high level of details about its design and it does not exist a thorough study of how to use the system in teaching and research. In fact, they have announced that the system is still under development. Another difference among IR-Base, IR Framework and IR-Components is that, whilst the first ones only provide a framework and a pool of components which the programmer has to plug in order to create the applications, the last one provides the components in a format that makes easier the creation and execution of IR applications.

Independently of these educational goals, classroom materials (text books, slides, problems, and so on) are another of the great needs of IR teaching. Jones regretted that lack in his inquiry-based learning approach [30]. He also noticed that feedback from students is another useful tool for improving the goals of the subject. The approaches of Henrich and Morgenroth [18] and Sacchanand and Jaroenpuntaruk [31] could be appropriate

for some courses too. However, the methodology presented in this paper tries to offer materials in the way of Croft et al. [7], but linking theoretical and technical concepts in a bit different and stronger way.

The technical-oriented IR methodology

The multi-disciplinary nature of IR almost invariably causes LIS researchers to focus on abstract or high-level matters, delegating the technical or low-level ones to computer scientists and engineers (e.g.: distributable systems, index compression techniques and so on). Hence, CS students have to be able to analyze the abstract requirements and transform them into programming code. The traditional way to make them familiarized with this procedure is to include some laboratory classes where students often have to program their own applications starting from an informal specification. Nevertheless, this approach gives rise to a couple of problems. On the one hand, a stronger association between theoretical concepts and programming code is needed in order to make this transition easier. On the other hand, students' time and effort should not be wasted in programming a complex and non-illustrative code in laboratory classes.

Another problem of the courses with a high level of technical detail is the search or development of sample applications. Although there are a lot of available and mature IR systems on the Internet, their code is often too complex to be explained to students. It would be better for teachers to have at their disposal a simpler custom system that illustrates the concepts at the appropriate detail level. At this point, teachers have to choose between a set of sample applications and a framework. Whilst the first alternative allows users to test immediately how the code works, the second one helps programmers to create or modify applications or components.

The problem of achieving a stronger linkage between theoretical concepts and the final programming code can be solved by means of the inclusion of the appropriate design diagrams and pieces of code in the course notes or slides. Even when the pedagogic materials become harder to maintain, this technique helps to create a better association between both parts of the subject. Further more, if the diagrams and the pieces of code are placed immediately after each theoretical concept or procedure, students will obtain a first aid guideline to solve each problem.

The aforementioned election between sample applications or a framework is especially interesting when the course notes or slides contain sample code. On the one hand, if students had at their disposal the complete application from where the code was taken, they could easily verify how it

works. On the other hand, a framework could guide them in the correct development of their coursework. Fortunately, it is possible to take advantage of all those features at the same time by means of the combination of both ideas, through a simple framework and a little set of implementations from whose fusion the sample applications will result. That will allow the teacher to choose either to provide the students only with the framework or to give them the full sample applications.

With regard to the coursework, an alternative to the problem of developing an application or a component from scratch could be to provide the students with a set of interfaces and a skeleton, making them develop the rest of the code. The skeleton would cover all the non-illustrative parts of the application or component they have to develop, whereas the guidelines offered by the set of interfaces would force students to follow a standard in the completion of the rest of the code. Orienting the framework to this structure of skeletons and interfaces will make the coursework more interesting for students.

Moreover, coursework evaluation could be easier for teachers too because following this approach students only have to implement a reduced and concrete number of classes specified by their corresponding interfaces, so their evaluation could also be reduced to a simple checklist about the number, versatility and efficiency of the implementations programmed by each student. The effectiveness of those modules would also become easy to check, since teachers would only have to plug them in their skeleton instead of their default implementations. Efficiency can also be checked by adding measuring techniques in the appropriate points of the skeleton.

In short, the approach presented in this paper consists of two main parts. The first one is helping to link theoretical concepts and software products, generally by means of the inclusion of diagrams and sample code in the course notes or slides and the second one is improving coursework selection, avoiding non-illustrative tasks in student's coursework. In order to follow this methodology, the teacher has to choose the right materials. A good selection can help with both points and make the evaluation easier.

The IR-Components framework is presented in this article as the pedagogic tool that will fulfil the requirements of the explained approach, so teachers will not have to create it from scratch. In addition, some of its features, like its modularity, will make it valid for other tasks, like the quick developing and testing of new IR techniques. Nevertheless, its educational purpose does not make it appropriate as a commercial system or as high-performing researching environment, capable of operating with large collections of documents and queries like .GOV2 [32].

IR-Components

The IR-Components project[1] has been designed as an object-oriented framework for developing IR applications. The entire software product has been developed using only the J2SE API of the Java programming language, XML files, the Apache log4j framework for log file generation [33] and the JUnit framework for unit test generation [34], so a programmer can read or modify all the code only knowing these four technologies. It is important to remark that the IR-Components project has been designed according to several design patterns, most of them introduced in order to create a robust and scalable architecture with pluggable components.

The IR-Components framework provides developers with a set of components which will have to be interconnected to build IR applications. This interconnection is a relatively easy task, since it only consists of one component invoking at least one of the APIs provided by another component. Internally, each of the components contains a pair of modules. On the one hand, programmers have the "API module", which offers them a set of interfaces and it can also offer the skeleton of an algorithm. Some of the aforesaid interfaces are APIs for externally invoking the functionalities of the component, whilst the rest are facades for completing the internal tasks delegated by the skeleton. On the other hand, they have the "implementation module", which consists of several classes that implement all the interfaces provided by the API module. The IR-Components project includes a default implementation module for each API module, but programmers could develop their own third-party implementations. The change of implementation would be very easy, since they would only have to choose their own classes in a configuration file. Figure 1 clarifies the structure of a component and how they are interconnected.

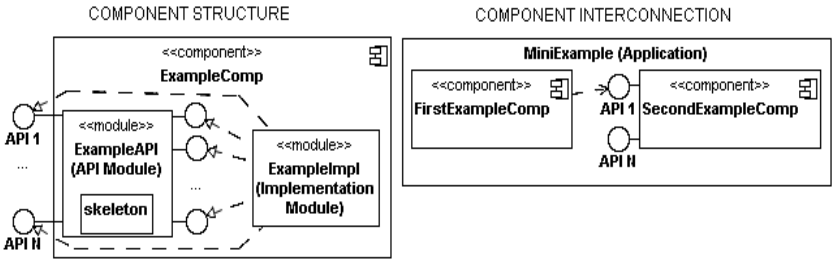

Fig. 1. Component structure and interconnection

[1] Current URL: http://www.tic.udc.es/~rlopezga/ir-components/index.html. User: "reviewer". Password: "tlir2010".

The naming convention used in this project is the following: applications are named according to the MiniXXX notation (e.g.: a searcher application should be called MiniSearcher). Components, which are part of applications, follow the XXXComp format (e.g.: a component containing the search engine of a searcher application should be called SearchEngineComp). Finally, for the modules that constitute a component, API modules follow the XXXAPI notation, whereas implementation modules are named according to the XXXImpl format (e.g.: a component called SearchEngineComp should be formed by the SearchEngineAPI and the SearchEngineImpl modules).

IR-Components could be used in at least three ways:

- To combine the provided components using their API and default implementation modules in order to quickly create an IR application.
- To generate a new implementation module for at least one component in order to change its behaviour. Then, the programmer should combine the new component with others in order to create a different application.
- To use the source code of a certain component to explain how it works.

Whilst the first use of IR-Components does not have much relevance for the CS staff, the second one is especially interesting for teachers who want to provide their students with a starting point for the coursework. In this case, it is recommended that a library be generated (e.g.: a jar file) with each of the API modules of the components, and then provide students with those libraries and their respective documentation in order to force them to create their own components through the development of the respective implementation modules. It is important to highlight that, as students do not possess the sources of default implementation modules, reverse engineering is not possible for them. The second use of IR-Components could also be interesting to programmers who want to quickly develop and test their new techniques, expecting the new component to be more efficient or versatile than the one generated using the default implementation. Finally, the third use could help teachers to explain with a higher level of detail how a certain component works.

The project components are particularly oriented to build three applications, MiniCrawler, MiniIndexer and MiniSearcher.

MiniCrawler is an implementation of the traditional web spider that uses a single thread to perform a breadth-first traversal of the target website and downloads the pages that match the specified file types, creating a data structure that stores the main attributes of the file. It also has support for the robots exclusion standard. The main educational objectives that have been identified for MiniCrawler are the following:

- To illustrate the breadth-first traversal in the crawling algorithm.
- To know how to access a URL and how to save its content (simple text file, compressed file or even by means of a database).
- To establish an access policy for URLs, taking into account several factors like their content type, whether they belong to the crawled website or not, whether they are excluded for robots or not and so on.
- To parse different kinds of files (plain text, HTML, XML and so on) in order to get new URLs to be accessed.

In order to achieve those aims, MiniCrawler only needs one component called CrawlerComp. The API module of this component is mainly formed by a skeleton (class Crawler) and some interfaces to which the skeleton delegates the illustrative parts of the algorithm, expecting to be coded in the respective implementation module. The diagram presented in Figure 2 shows this organization.

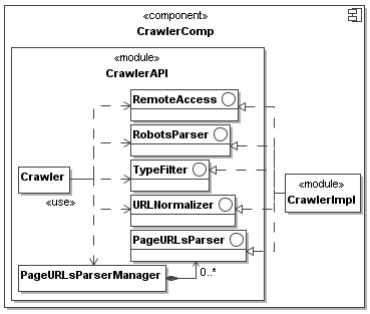

Fig. 2. Architecture of MiniCrawler and its components

The first educational objective is covered by all the classes in general, but the skeleton provided by the class Crawler is the most important contributor. The second objective is fulfilled by the RemoteAccess interface. The third one is reached by means of the RobotsParser, TypeFilter, and URLNormalizer interfaces, which provides methods to check the robot exclusion standard, the content type of the URLs and its belonging to the crawled website respectively. Finally, the fourth objective is covered by the PageURLsParser interface and the PageURLsParserManager class, which is a pool of PageURLsParser objects.

Each of those interfaces can be implemented following several strategies. For example, a programmer can create an implementation of the TypeFilter interface either based on the file extension (efficient but not valid in every case) or the MIME type offered by an HTTP request (more

correct but inefficient). In the default implementation module, the developers chose a class based on the file extension.

The purpose of MiniIndexer is to create a document index and a term index for a document collection. The application has at its disposal different ways to store the indexes in memory and on disk (plain text, Java serialized objects and so on) and it also supports as many MIME types as plug-and-play parsers are installed in the system (plain text, HTML and XML documents by default). What is more, MiniIndexer also supports some other term processing advanced configurations, like the stop words mechanism, the list of characters that have to be treated as separators (e.g.: hyphen, plus sign, etc.) and the list of special characters that may be transformed into others (e.g.: those which contain tilde, dieresis, etc.). The primary teaching goals that have been identified for MiniIndexer are:

- To illustrate an indexing algorithm.
- To extract the tokens of various documents with different structures.
- To be conscious of the problems brought by some special characters that should be normalized and some others that should act as separators.
- To determine which tokens are significant enough to be included in the term index, generally by means of the stop words technique.
- To experiment with several data structures, file formats and loading and storage techniques for the index.

The architecture and components of MiniIndexer are shown in Figure 3.

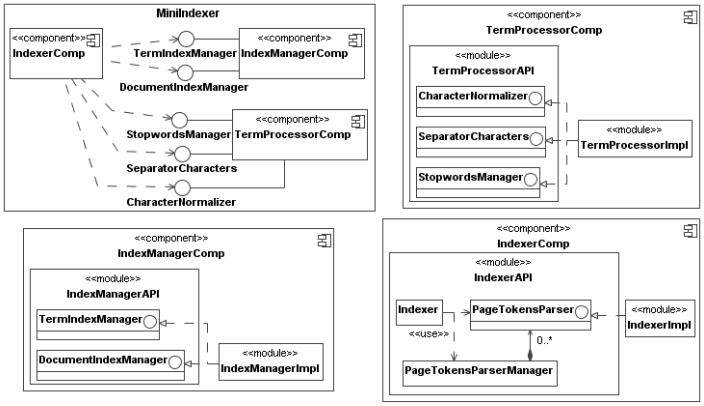

Fig. 3. Architecture of MiniIndexer and its components

The component of MiniIndexer that contains the skeleton of the indexing algorithm is IndexerComp. In order to improve extensibility, it detaches from the index format and operations, delegating this responsibility

to a component called IndexManagerComp, which decides the data structures to be used, the file format and the loading and storage strategies (cache, etc.). The term processing techniques are also encapsulated in a component called TermProcessorComp.

The API module of IndexerComp is principally formed by the aforementioned skeleton (class Indexer) and some more classes and interfaces. The skeleton delegates the illustrative parts of the algorithm to those classes and interfaces and also to the other components that constitute the MiniIndexer application, covering the first educational aim. The PageTokensParser interface and the PageTokensParserManager class, which acts as a pool of PageTokensParser objects, are in charge of reaching the second teaching goal of the application.

The second component, TermProcessorComp is only constituted by a set of APIs for term processing and their respective implementations. It lacks a template for an algorithm. The CharacterNormalizer and SeparatorCharacters interfaces and their implementations are responsible for covering the third educational goal, whereas the StopwordsManager interface is in charge of the fourth teaching aim.

Finally, the third component, IndexManagerComp, only provides a mechanism of APIs and implementations for term and document index managing (creation, update, lookup, load and storage). The DocumentIndexManager and TermIndexManager interfaces and their respective implementations fulfil the fifth pedagogic objective of MiniIndexer.

The versatility of MiniSearcher lies in its capability of being configured to act as a standalone search engine or as a broker of a distributed environment in combination with multiple query server instances. In any case, the application accepts multiple user interfaces (interactive console, batch process and window interface); multiple weighting models and the same term processing techniques as MiniIndexer (stop words, separator character configuration and so on). On the other hand, when MiniSearcher is acting as a distributed environment, it supports broker hierarchy, asynchronous result reception, several transport protocols (e.g.: TCP and UDP with and without multicast), multiple types of data encoding to send the results through the network, and multiple algorithms to get the best results at the broker. The pedagogic objectives of MiniSearcher are the following:

- To illustrate the overall process of query solving in centralized and distributed environments.
- To evaluate the usability of the different types of user interfaces.
- To study in depth the advantages and problems brought by normalized characters, separator characters and stop words.
- To detail the functioning and features of the different weighting models.

- To implement a result sorting algorithm and to evaluate its performance.
- To implement the communications between brokers and query servers and to learn the consequences of using different transport protocols.
- To investigate the advantages and disadvantages of the different encodings for sending the data through the network.
- To know the problem of result selection in distributed environments.

The components of MiniSearcher have to be interconnected in different ways depending on the scenario in which the application is running (standalone, distributed with a broker and some query servers or distributed with a complete hierarchy of brokers and query servers). The simple change of a parameter in a configuration file is sufficient to change from one to another. Every scenario uses some more components than the previous one, so the standalone scenario where the user interface directly connects the search engine is the simplest one. Figure 4 clarifies that architecture.

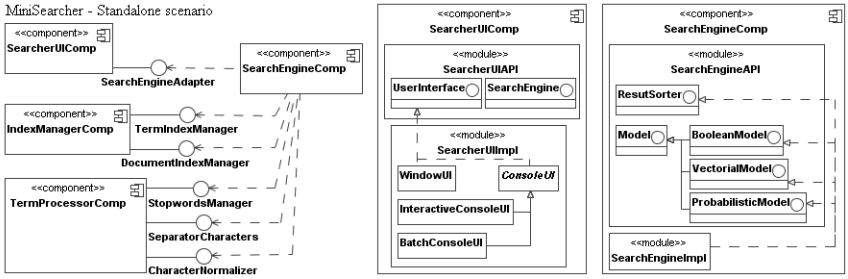

Fig. 4. Architecture of MiniSearcher in a standalone scenario

As some needs of MiniSearcher are the same as in other applications above, the programmer can re-use some components from them. For example, as a search engine needs to lookup the indices in order to solve the queries, the IndexManagerComp used in MiniIndexer is now re-used in MiniSearcher. Other components like TermProcessorComp can also be re-used, but as the term processing can be done in several places of the searcher, the programmer must decide where it is going to be placed. This is how the third pedagogic aim is covered.

The first new component is the user interface (SearcherUIComp), which covers the second teaching goal of the application and provides an API for the possible user interfaces and some classes that could be used in its implementations. Most of those classes are graphical components, with the exception of SearchEngineAdapter, that communicates the user interface with the search engine. In order to prove that the API can be used by con-

sole user interfaces and graphical user interfaces, the default implementation of the module consists of an interactive console, a console that executes a batch process and a window environment.

The search engine is also independent from the rest of the application, so it is confined in a component called SearchEngineComp. As it is shown in Figure 4, the component has a set of interfaces for the different weighting models supported by the engine (Model, BooleanModel, VectorialModel and ProbabilisticModel). This way, the fourth teaching goal is covered. There is another interface for result sorting (ResultSorter). Its implementations are responsible for covering the fifth teaching goal.

The second and third scenarios consist of a distributed architecture. Figure 5 presents a simple schema of both scenarios. In the second one, there is only a root broker (RB) and a variable number of query servers (QS). In the third one, there is a hierarchy composed of a root broker (RB), some intermediate brokers (IB) and the query servers (QS) which are the leaf nodes of the tree. Figure 6 studies the architecture of the third scenario, since the second one is a particular case of the third without any IB.

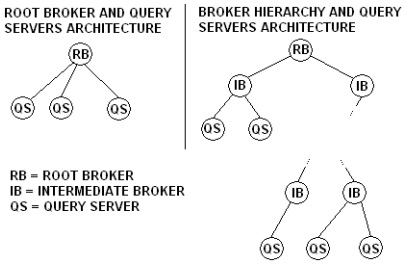

Fig. 5. Distributed scenarios for MiniSearcher

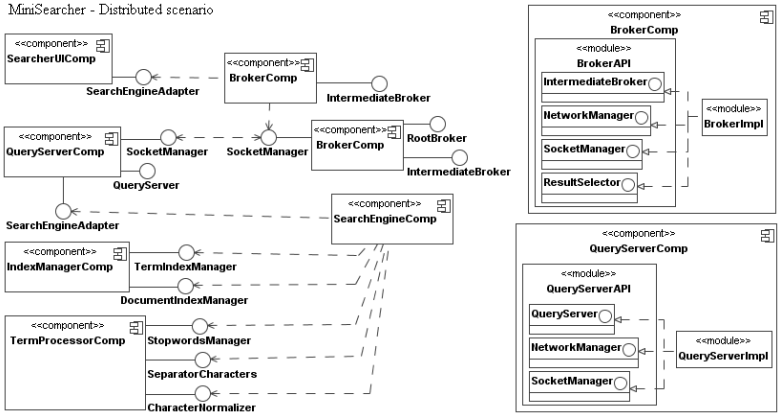

Fig. 6. Distributed scenarios for MiniSearcher

In order to develop both scenarios, two new components are needed: BrokerComp and QueryServerComp.

BrokerComp provides another implementation of the SearchEngineAdapter facade from the SearcherUIComp when it acts as the root broker; and it uses its own IntermediateBroker interface when it works as an intermediate broker. In both cases, the SocketManager, NetworkManager and ResultSelector interfaces are in charge of developing the sixth, seventh and eighth teaching goals respectively.

QueryServerComp offers an API (QueryServer interface) for component invocation, and it also offers the SocketManager and NetworkManager interfaces, which are responsible for reaching the sixth and seventh teaching goals respectively from the query server's side.

Assessment and feedback

The first time that Professor Fidel Cacheda taught a subject about IR at the University of A Coruña (Spain), it was in a master degree in the academic year 2007-2008. Professor Cacheda and some other contributors related their experience in [4]. Their methods were focused on the decomposition of the different typical problems of IR and on the strategies to implement the low-level solutions. They also enumerated some inconveniences of their "Internet Information Retrieval" subject and its practical coursework. As a solution to the problems, Cacheda proposed some "live-programming" classes and the possibility to offer the students a basic implementation of the subsystems that compose the coursework (crawler, indexer and searcher) as a basic starting point.

For the next course, Cacheda changed the format of the practical coursework, using the first version of IR-Components as the aforementioned starting point. In particular, the students were provided with a set of binary libraries containing the API modules of the framework and they were told to create all the corresponding implementation modules with the exception of the searcher's user interface and the distributed environment. They also were given the documentation associated to those modules, so their first task was learning how to use the libraries.

In order to compare both years' results, students took an anonymous opinion poll about the course. In basic outline, they had to give their opinion about the difficulty of implementing each of the proposed components and the contribution they entailed to the increase of their knowledge. In addition, the poll also included some questions to assess the contribution of using some other pedagogic resources (design diagrams, software

documentation, the coursework forum and so on). Some of the questions consisted of rating something between 1 and 10 and some others asked for an extended explanation. In both the 2007-2008 and 2008-2009 academic years 7 students were called to answer the questions, but in both cases only 6 of them took part in the poll.

In the general section about the coursework, students had to rate the workload and the difficulty. Numerical results are shown in Table 1.

Table 1. Coursework general results

	Score '07-'08	Score '08-'09	Ratio of variances	Significance
Workload	7.5	7.57	1.5906	> 97%
Difficulty	6.7	6.43	1.7427	> 99%

In the additional comments, one of the students asked for some more techniques to be applied (e.g.: a ranking algorithm like HITS or PageRank). Another student suggested starting the coursework with some parts already implemented, alleging that they were too tedious to be implemented by the students and their contribution to his/her knowledge was insufficient. Nevertheless, a third one suggested exactly the opposite, in other words, that all the coursework was interesting and it should be mandatory to implement the three applications from scratch. A fourth student compared the workload of making the applications in Python and in Java, and he suggested that it is easier to implement them in the first programming language.

Even though the global rates did not change significantly for the second year, everybody agreed that having the IR-Components APIs as a starting point is clearly an advantage. The main reason given by the students was that they avoided a lot of tedious work. Unnecessary complexity was removed and the students could concentrate on the most important parts of the engine. Most of them also agreed that the main inconvenience of this kind of coursework is that some parts of the software act as a black box, making it more difficult to understand in the beginning, but they also clarified that the design diagrams and the documentation are sufficient to overcome this initial problem. In addition, every student of the second course declared that the workload is sufficient, but 5 out of 6 affirmed that a PageRank module could be a good extra exercise for the next course since it is historically important in IR and it is the only part of the theoretical contents that had not been reflected on the practical coursework.

Regarding the different resources offered to the students, in both academic years everybody agreed that the forum is convenient for sharing some information about the development of the applications. However,

they also agreed that personal attention is also necessary in some cases, especially for the students of second year whose interaction with an unknown API makes them more prone to technical problems. The acceptance of some other resources was remarkable too. Some of these materials, such as hints to help students to solve some programming problems, were available in both years, whilst the others, programming documentation and design diagrams, were only available to the second year of students. Numeric results are shown in Table 2:

Table 2. Statistics about teaching resources

Resource	2007-2008	2008-2009
Forum	7.67	8.57
Hints	8.6	9.57
Programming documentation	N/A	9.33
Design diagrams	N/A	9.33

In spite of the fact that there were only seven students in each promotion, four thorough statistical tests were made in order to analyze the average and the variance of the difficulty of implementing each feature and the contribution that they entailed to the knowledge of the students. Unfortunately, the lack of a sufficient number of samples determines that no conclusion can be reached when the average of the different factors is studied.

More interesting conclusions can be reached when the variance is studied. Table 1 shows the ratio of variances for the difficulty and the contribution of the coursework in both groups. As in both cases the variance is significantly smaller for the second year and the analysis of the p-value also shows a high degree of certainty for this hypothesis, the first conclusion extracted from the statistical study is that the application of the new methodology reduces the number of students who think that the coursework is too difficult or easy, or that it contributes a lot or nothing at all to their knowledge. Even when decreasing the third of the four groups too is a bad partial result, the combination of the previously presented methodology and materials constitute an improvement in the subject.

On the teachers' side, the opinion is quite different. In small and medium scale projects like this, working with new APIs and continuing other people's work is more difficult than starting a project right from scratch, so coursework in the second academic year was harder. However, the difficulty did not change significantly in the poll. Also, the number of questions asked by the students was significantly bigger during the second year, so the teachers think that the students were more interested in the second year's coursework and this helped them to overcome difficulty.

Conclusions and future work

This paper has presented a technical approach to teaching IR which could result interesting mainly for teachers belonging to the CS curricula. As the global idea lies on helping students to make the transition from theoretical concepts and specifications to final software products, this methodology is based on a strong linkage between theoretical contents and programming code, not reducing its utilization to laboratory classes and practical coursework, but including it in the classroom materials (notes, slides and so on). In addition, this methodology also focuses on other problems like the coursework evaluation and the difficult choice of sample applications, since there is a lack of appropriate materials.

This article has also submitted IR-Components, a multi-module object-oriented software product which combines the ideas of an IR framework and the common IR sample applications. The project has been studied in depth in order to show how several pedagogic objectives of IR subjects could be fulfilled by means of its use. Moreover, this project could be useful not only for teachers, but also for programmers who are in need of a simple starting point to test their new techniques and algorithms.

This article has also evaluated the difference between using and not using the aforementioned methodology and materials. To that end, the students of two years of a post-graduate Internet Information Retrieval class took an anonymous opinion poll. The results given by the students suggest three interesting conclusions. First of all, the availability of a wide range of resources (discussion forum, programming documentation, design diagrams, source code) and their integration with theoretical concepts is really useful to improve students' interest and understanding. These resources are really necessary for the development of their coursework. Secondly, the availability of an IR framework as a starting point for students' coursework helps them to avoid tedious and non-illustrative parts of the coursework and it also guides them to the correct design of IR applications. Finally, the approach described in this paper significantly reduces the variability of their opinions about the difficulty and the contribution of the coursework. Hence, the number of students who think that the coursework is too difficult or easy or that it makes too big or to small contribution to their knowledge is reduced.

On the teacher's side, the usage of IR-Components makes easier to evaluate students' work, since they have only to check which of the proposed functionalities were implemented and how they were programmed.

However, it is clear that the full fruition of this methodology will be reached only after several years of application, so it will be necessary to

continue improving and adapting it, generally by means of the feedback submitted by teachers, students and researchers. The immediate future work of the general approach is, mainly, to improve the way of linking theoretical concepts and the final programming code. This will mean more experiments in this respect and a more thorough monitoring in student's learning process.

Regarding to IR-Components project, two different paths will be followed in the future. The first one will consist of adding new features to the framework with two aims in mind: to include some IR concepts which can be interesting for students and to make IR-Components become not only an educational framework, but also a research tool. As a first step, the next versions will include the IR concepts that the students suggested in the opinion poll (e.g.: PageRank), some other interesting ones (e.g.: query expansion, stemming, more weighting models, etc.) and some wrappers for a few research engines, like MG4J [35] and Terrier [13].

The other path to follow is to research a way to extend IR-Components in order to help teachers and students of other branches (e.g.: LIS). The application resulting from this extension would consist of a canvas where the user could create a search engine without editing a single line of programming code, but simply dropping and interconnecting some visual representations of several parts of IR-Components. This would allow students of non-technical branches of IR to be introduced in the technical concepts at a level of abstraction appropriate for their particular studies.

Acknowledgements

This work was partially supported by the Spanish Government under project TIN 2009-14203, the European Social Fund and the Dirección Xeral de Ordenación e Calidade do Sistema Universitario de Galicia of the Consellería de Educación e Ordenación Universitaria - Xunta de Galicia (Spain).

We would also like to thank Professor Fernando Bellas for his educational guidance on the architecture of IR-Components.

References

1. The joint task force on computer curricula IEEE-CS & ACM (2001, 2005) Computing curricula 2001 and Computing curricula 2005. Computer science. http://www.acm.org/education/curricula-recommendations

2. Ruthven I, Elsweiler D and Nicol E (2008) Designing for users: A holistic approach to teaching information retrieval. In: Proc. of the 2nd international workshop on teaching and learning of information retrieval. http://www.bcs.org/server.php?show=ConWebDoc.22356
3. Thornley C (2008) Teaching information retrieval (IR) as a philosophical problem. In: Proc. of the 2nd international workshop on teaching and learning of information retrieval. http://www.bcs.org/server.php?show=ConWebDoc.22354
4. Cacheda F, Fernández D and López R (2008) Experiences on a practical course of web information retrieval: Developing a search engine. In: Proc. of the 2nd international workshop on teaching and learning of information retrieval. http://www.bcs.org/server.php?show=ConWebDoc.22357
5. Robertson SE, Sparck Jones K (1997) Simple, proven approaches to text retrieval. Cambridge Technical Report. http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-356.pdf
6. Manning CD, Raghavan D and Schütze H (2008) Introduction to Information Retrieval. Cambridge University Press.
7. Croft B, Metzer D and Strohman T (2009) Search Engines: Information Retrieval in Practice. Addison Wesley.
8. Freed N, Borenstein N (1996) Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies. RFC 2045, 31 pages. http://www.ietf.org/rfc/rfc2045.txt
9. Koster MA Standard for Robot Exclusion. http://www.robotstxt.org/orig.html
10. Porter MF (1980) An algorithm for suffix stripping. In: Program, vol. 14, no. 3 pp. 130-137. http://tartarus.org/~martin/PorterStemmer/def.txt
11. Zobel J, Moffat A (1998) Exploring the similarity space. In: SIGIR Forum.
12. Apache Lucene: http://lucene.apache.org/
13. TERabyte RetrIEveR: http://ir.dcs.gla.ac.uk/terrier/
14. Calado P, Cardoso-Cachopo A, Oliveira A (2007). IR-BASE: An integrated framework for the research and teaching of information retrieval technologies. In: Proc. of the 1st international workshop on teaching and learning of information retrieval. http://portal.acm.org/citation.cfm?id=1181901.1181949
15. Mooers CN (1951) Making information retrieval pay. In Boston: Zator Co.
16. Zhu L, Tang C (2006) A module-based integration of Information Retrieval into undergraduate curricula. In: Journal of Computing Sciences in Colleges, vol. 22, no. 2, pp. 288-294. http://portal.acm.org/citation.cfm?id=1181901.1181949
17. Fernández-Luna JM, Huete JF, MacFarlane A, Efthimiadis EN (2009) Teaching and Learning in Information Retrieval. In: Information Retrieval. http://www.springerlink.com/content/drx6741t28k71612/fulltext.pdf
18. Henrich A, Morgenroth K (2007) Information retrieval as e-learning course in German—Lessons learned after 5 years of experience. In: Proc. of the 1st international workshop on teaching and learning of information retrieval. http://www.bcs.org/server.php?show=ConWebDoc.8765
19. Efthimiadis EN and Hendry DG (2005) Search engines and how students think they work. In: Proc. of the SIGIR conference. pp. 595-596. http://portal.acm.org/citation.cfm?id=1076034.1076145

20. Hendry DG and Efthimiadis EN (2008) Conceptual models for search engines. In: Spink A, Zimmer M (Eds.), Web searching: Interdisciplinary perspectives. Springer. pp. 277-307.
21. Herrera-Viedma E, Alonso S, Cabrerizo FJ, Lopez-Herrera AG, Porcel C (2007) A software tool to teach the performance of fuzzy IR systems based on weighted queries. In: Proc. of the 1st international workshop on teaching and learning of information retrieval. http://www.bcs.org/server.php?show=ConWebDoc.8767
22. Goharian N, Grossman D, Raju N (2004) Extending the undergraduate computer science curriculum to include data mining. In: Proc. of the international conference on information technology: Coding and computing (ITCC'04) pp. 251-254.
23. Johnson F (2008) On the relation of search and engines. In: Proc. of the 2nd international workshop on teaching and learning of information retrieval. http://www.bcs.org/server.php?show=ConWebDoc.22355
24. Airio E, Sormunen E, Halttunen K, Keskustalo H (2007) Integrating standard test collections in interactive IR instruction. In: Proc. of the 1st international workshop on teaching and learning of information retrieval. http://www.bcs.org/server.php?show=ConWebDoc.8783
25. Efthimiadis EN, Freier NG (2007) IR-Toolbox: An experiential learning tool for teaching IR. In: Proc. of the SIGIR conference (p. 914).
26. Jinguji D, Lewis W, Efthimiadis EN, Minor J, Bertram A, et al. (2006) The University of Washington's U WCLMA QA system. In: The 15th Text REtrieval Conference (TREC 2006) proceedings.
27. De Campos LM, Fernández-Luna JM, Huete JF, Romero AE (2007) A flexible object-oriented system for teaching and learning structured IR. In: Proc. of the 1st international workshop on teaching and learning of information retrieval. http://www.bcs.org/server.php?show=ConWebDoc.8769
28. Chau M, Huang Z, Chen H (2003) Teaching key topics in computer science and information systems through a web search engine project. In: ACM Journal of Educational Resources in Computing, vol. 3, no. 3, article 2. http://portal.acm.org/citation.cfm?doid=1029994.1029996
29. Wade S, Braekevelt P (1994) IR framework: An object-oriented framework for developing information retrieval systems. In: Program-Automated Library and Information Systems, vol. 29 no.1, pp. 15-29.
30. Jones GJF (2009) An inquiry-based learning approach to teaching information retrieval. In: Information Retrieval, vol. 12, no. 2, pp. 148-161. http://www.springerlink.com/content/kh60168006p30x63/fulltext.pdf
31. Sacchanand C, Jaroenpuntaruk V (2006) Development of a web-based self-training package for information retrieval using the distance education approach. In: The Electronic Library, vol. 24, no. 4, pp. 501-516.
32. .GOV2 collection: http://ir.dcs.gla.ac.uk/test_collections/gov2-summary.htm
33. Apache log4j: http://logging.apache.org/log4j
34. JUnit: http://www.junit.org/
35. Managing Gigabytes for Java (MG4J): http://mg4j.dsi.unimi.it/