# IR-BASE: An Integrated Framework for the Research and Teaching of Information Retrieval Technologies

**Pável Calado**
IST/INESC-ID
Av. Prof. Cavaco Silva, 2780-990 Porto Salvo, Portugal

**Ana Cardoso-Cachopo**
IST/INESC-ID
Av. Rovisco Pais, 1, 1049-001 Lisboa, Portugal

**Arlindo L. Oliveira**
IST/INESC-ID
R. Alves Redol, 9, 1000-029 Lisboa, Portugal

**Abstract**

**Due to the rapid growth in digital storage technologies, Information Retrieval (IR) has gained a significant importance, both for academia and for the industry. However, very few proposals have been made for the creation of an environment capable of both providing useful IR services and acting as a workbench for the design, implementation, and research of new IR solutions. In this work, we propose IR-BASE, a basic object oriented framework for the integration of components, documentation and services, focused on the rapid development of prototypes for research and teaching. At its core, IR-BASE consists of a set of base classes that provide a skeleton to create new IR components, a set of guidelines to ensure that the developed components are fully interoperable, and a component pool, where IR-BASE users can find implementations of all kinds of functionalities needed to build a full IR system, and to which IR-BASE users can contribute with their own implementations. IR-BASE is of interest not only to IR researchers and professionals, who want to rapidly develop prototypes to test new ideas, but also to teachers and students, who will have an easily configurable, modifiable and extendible set of components to act has a basis for learning, building and experimenting with current IR algorithms.**

*Keywords: IR systems, components, development, open-source*

## 1. INTRODUCTION

Information Retrieval (IR) concerns the study and implementation of techniques for the storage, treatment, and access to digitally stored information. Due to the rapid growth in digital storage technologies, and to the popularization of the World Wide Web, IR has recently gained a significant importance, both for academia and for the industry. However, and although IR research has been carried out for more than fifty years, few proposals have yet been made for the creation of an environment capable of, not only providing useful IR services, but also of acting as a platform for the rapid design, implementation, and research of generic IR solutions.

To fill this gap, we introduce the IR-BASE project. IR-BASE aims at creating an environment for integrating tools, documentation and services to help IR teaching and research. The core of such environment is a set of base classes that provide a skeleton to create new IR components, a set of guidelines to ensure that the developed components are fully interoperable, and a component pool, where IR-BASE users can find implementations of all kinds of functionalities needed to build a full IR system, and to which IR-BASE users can contribute with their own implementations.

The emphasis of IR-BASE is on modularity and interoperability. The IR-BASE framework facilitates the construction of specific information processing components that can later be fitted together to build full IR systems. This component-based methodology has several advantages. For researchers, it can provide easily reusable implementations of functionalities that are common to many different IR research problems (e.g., text indexing). For industry developers, it can provide an easy way of testing many different implementations of a specific functionality, by building system prototypes where each component can be easily replaced. For teachers and students, it allows both a high-level view of an IR system, where it is seen as a set of different communicating components, and a low-level view, where the internal workings of each component can be independently analyzed. Using IR-BASE, students will have the opportunity to build and learn about IR systems by either connecting, configuring, and modifying existing components, or by programming their own components from scratch.

Although the IR-BASE project is still in its startup phase, the proposed framework has already been specified and its implementation has begun. The project will be developed under an open-source philosophy, where anyone can contribute by providing their own implemented components. In fact, implementation of the basic IR-BASE infrastructure should be a quick process and IR-BASE itself will grow together with the user's contributions, with new components and documentation being regularly added.

In the short term, this project should provide an important help in IR research and teaching and an effective basis for the exploration of different IR solutions. In the long term, we expect IR-BASE to contribute to the integration of teaching and research efforts and to the cooperation between schools and research groups worldwide, by providing an effective way of exchanging and sharing implementations of different IR solutions.

In the following sections, we introduce and explain the ideas behind the IR-BASE framework. This paper is organized as follows. Section 2 describes and analyzes prior work, similar or related to the IR-BASE proposal. Section 3 specifies and discusses the IR-BASE framework. Section 4 comments on how IR-BASE can be applied to IR teaching and learning. Finally, Section 5 concludes our discussion and presents future steps to be taken within the project.

## 2. RELATED WORK

Although IR-BASE is not an IR system, many such systems have been proposed with similar goals in mind. A classic example is SMART [3], a system developed in the 1960's at Cornell University. SMART was designed as an IR research tool and, besides a set of common text processing functions, it can be expanded through programmable modules.

Another well known example is the MG system, which was also designed as a research tool. MG is an implementation of the algorithms described in the book by Witten et al. [10], which emphasizes the use of text indexing algorithms capable of dealing efficiently with a large amount of data through, for instance, compression techniques.

More recent projects, such as Terrier [7] or Lemur [5], have come closer to the proposal of this paper. Terrier is a modular search system built for experimentation. It provides an API to access and extend its many information processing modules and currently implements a great variety of algorithms. Lemur has a similar philosophy, although it is more dedicated to the implementation of statistical language models [8].

Finally, systems like Lucene [6] or WIRE [9], are also worth mentioning. Although being good examples of publicly available, open source, IR systems, these generally aim at being fully functional and efficient, and not so much at the ability to be modified and expanded.

When compared to any of these works, the IR-BASE project has a much wider goal. First, as mentioned before, IR-BASE is not an IR system, but a framework to build interoperable IR system components. Thus, unlike previous works, IR-BASE provides a convenient solution for the integration of different modules, even if these come from different authors and have been designed to work with different systems. Second, these systems are mostly focused on a single IR task--searching. IR-BASE is a generic framework and should be useful to developers working on many IR problems, such as classification, filtering, clustering, or any other [1]. Finally, the component-based design of IR-BASE allows IR teachers and students to easily create, modify, or simply observe the workings of a full IR system, or of any of its components, thus providing a practical workbench for IR teaching and learning.

A project similar to IR-BASE, in the sense that it shares the more general goal of providing an infrastructure to develop IR systems, is that of the Alvis Consortium [4]. The Alvis Consortium conducts research in the creation of topic-based independent search engines that should interact through a federated network. Unlike IR-BASE, however, the Alvis project focuses on peer-to-peer networks and communication standards as a means of interoperability. Further, the search engines participating in these networks should be capable of using semantic web technologies [2], a restriction not present in IR-BASE.

## 3. THE IR-BASE FRAMEWORK

In this section, we explain how an IR system can be built under the IR-BASE framework. In our discussion, we will use the term *component developer* to refer to the person that develops individual components under the IR-BASE framework. We will use the term *system developer* to refer to the person that uses the IR-BASE framework, and its components, to build a full IR system. We start by discussing the central idea behind the proposed framework: IR-BASE components.

## 3.1 IR-BASE Components

Components are parts of an IR system. In IR-BASE, they can be seen as independent sub-systems, that receive data, process it, and produce some form of output. The internal functioning of the components can be freely specified, to allow the implementation of any type of system. The interfaces of the components, however, are restricted, to ensure that these can be integrated to other components.
Components can have four types of interface: (1) an input interface, (2) an output interface, (3) a control interface, and (4) a probing interface. An input interface is a set of functions used by the component to obtain the data needed for processing. It provides a layer of abstraction separating the component's task from the format of its input data. The output interface is a set of functions that provide low-level access to the results of the component's work. It provides a layer of abstraction separating the actual results produced by the component from their internal structure. The control interface is a set of functions used to control the execution and parameterization of the component. The component itself is, in fact, an implementation of its control interface. Finally, the probing interface is a set of functions that access the internal workings of the component. It is used to have runtime access to the component's variables. This can be useful to observe the internal workings of a component and provide, for instance, live demonstrations that can be used in class by IR teachers. It is implemented by the component developer.

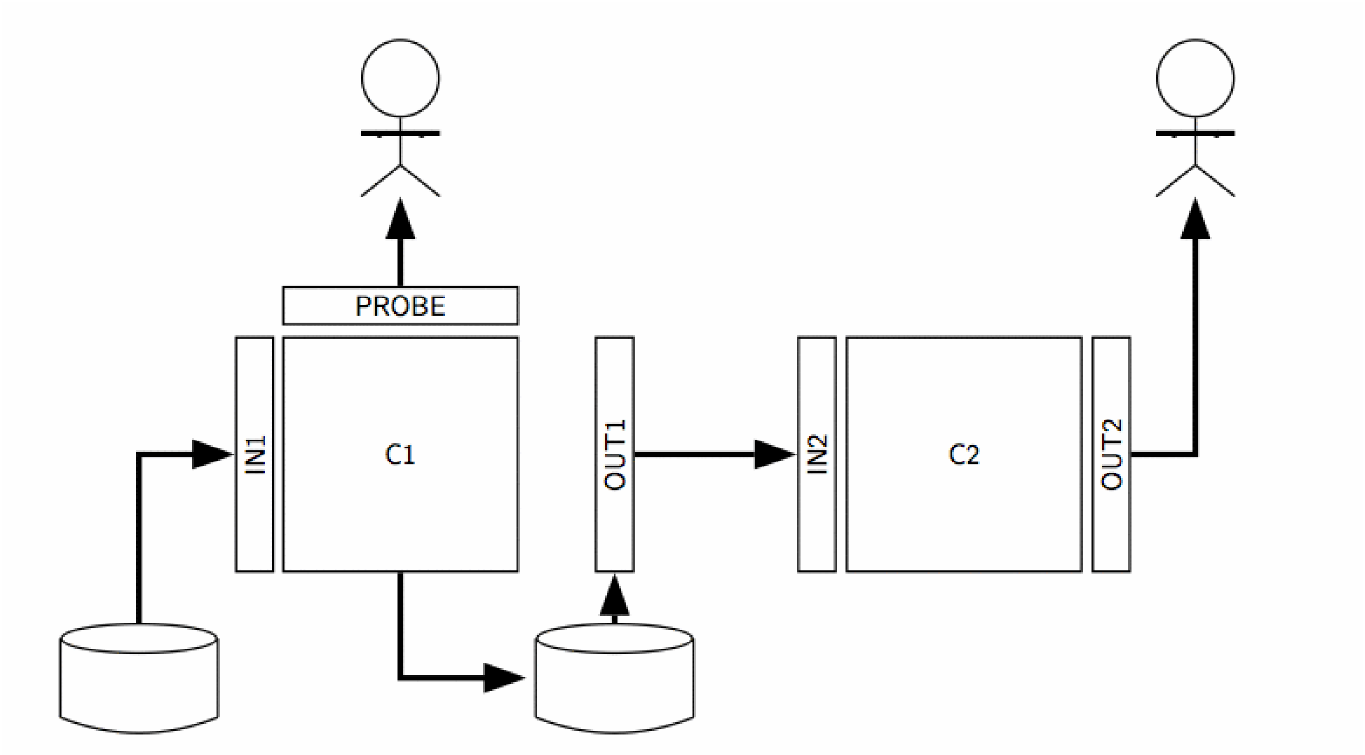

**Figure 1:** Example of an IR system built with IR-BASE components.

Figure 1 shows a schematic for how components can be organized to build an IR system. In this example, there are two interacting components, *C*1 and *C*2. To make this example more concrete, we can think of component *C*1 as a text indexer and component *C*2 as a document ranking module.

The input interface of component *C*1, *IN*1, implements functions that read text files from disk one word at a time. Component *C*1 uses these functions to read, parse, and index a set of documents. The component developer should define these functions, but not implement them. A system developer who, for instance, wishes to build an IR system that indexes PDF documents using *C*1, is responsible for providing the appropriate implementation of its input interface. A different system developer, who wishes to build an IR system that parses HTML documents, should provide another implementation of this interface.

The index produced by *C*1 is then written to disk. The developer of *C*1 also implemented the output interface *OUT*1 . Interface *OUT*1 might, for instance, contain a function that provides the inverted list of a given term. The input interface of component *C*2, *IN*2, uses the functions in *OUT*1 to obtain the inverted lists. This data is then processed by *C*2 and, given a user query, a ranked set of documents is produced. This set of documents is made available to a user through, for instance, a dynamically created Web page, whose script uses the functions implemented by the output interface *OUT*2.

The control interface of *C*1 might contain functions to start indexing the documents or setting parameters like the index compression level, for instance. The control interface of *C*2 might contain functions to retrieve ranked sets of documents, given a user query, or to set parameters like ranking thresholds, for instance. For simplicity, the control interfaces are not shown in the figure. Finally, the probing interface attached to *C*1 can provide, for example, a function that indicates how many terms have been already indexed. This information can be printed on screen while the indexer is running.

### 3.2 Creating IR-BASE Components

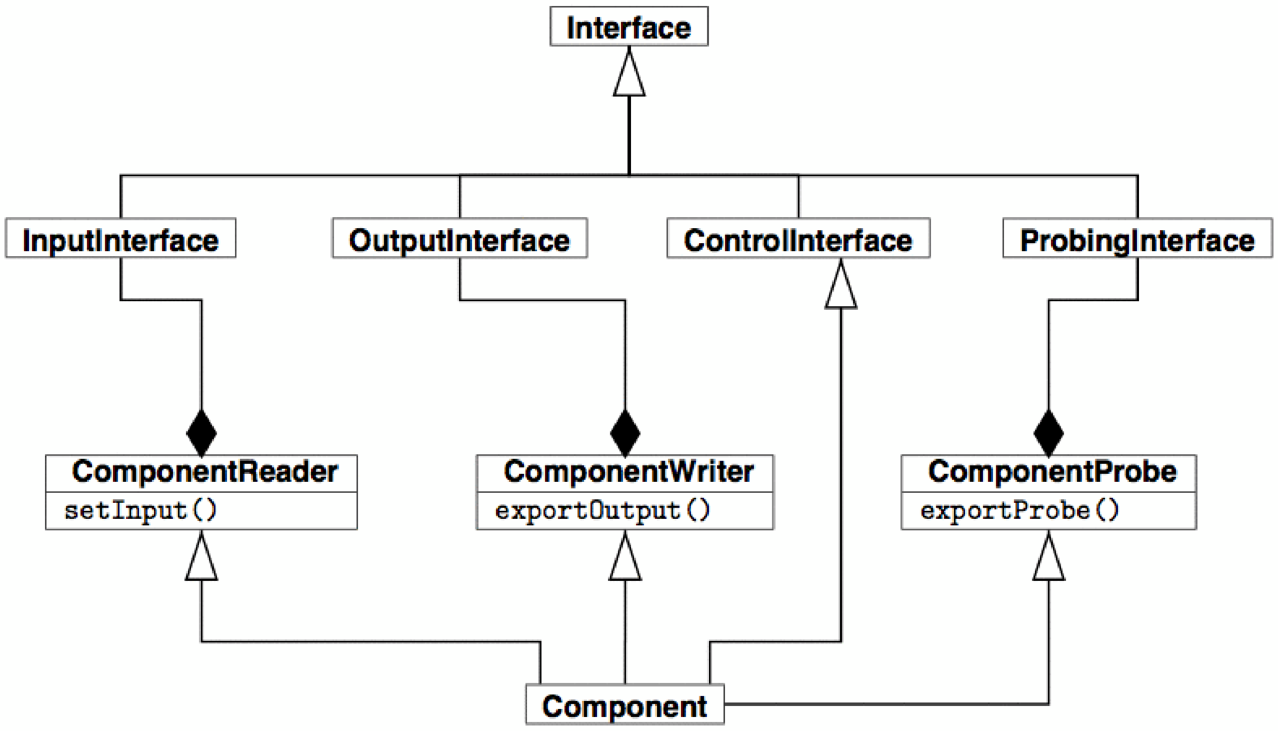

**Figure 2:** IR-BASE base classes.

Figure 2 shows a diagram for the class structure of IR-BASE components. The implementation of an IR-BASE component should follow this pattern. The *Interface* class represents a component interface, which can be of four different types: *InputInterface*, *OutputInterface*, *ControlInterface*, and *ProbingInterface*, as explained in Section 3.1.
The classes *ComponentReader*, *ComponentWriter*, and *ComponentProbe* represent the three possible characteristics of a component regarding its interfaces. A component that is also a *ComponentReader* contains an object of type *InputInterface*, which it uses to read the data for processing. A component that is also a *ComponentWriter* contains an object of type *OutputInterface*, which it uses to make the product of its operations available. This component can export its output interface so that other components, or the IR system, can use it. Finally, a component which is also a *ComponentProbe* contains a variable of type *ProbingInterface*, which it uses to make its internal state visible. This component can also export its probing interface so that other components, or the IR system, can use it.
The *Component* class represents a standard component. It can inherit the properties of *ComponentReader*, *ComponentWriter*, and/or *ComponentProbe*, depending on whether it needs an input, output, or probing interface, respectively. This class is always derived from a *ControlInterface* class, meaning that it inherits all the methods that control the execution and parameterization of the component.
In general terms, to create a new component, a component developer must (1) create new classes for each of the needed basic interfaces, and (2) create a new class for the component itself. For the input interface class, the component developer must determine the methods needed by the component to obtain the data for processing. For the output interface class, the component developer must determine the methods needed to access the data produced by the component. For the probing interface class, the component developer must determine the methods needed to provide a view of the component internal state. Finally, the component developer can create a new class for the control interface, containing the methods that will be used to operate the component.

### 3.3 The Component Pool

The IR-BASE component pool is a repository of components and documentation. It provides services for searching components, submitting components and submitting different implementations of interfaces to the existing components. When creating an IR system, developers should be able to search the component pool for components that might be useful. If an appropriate component is found, the developer will also have access to all existing implementations of its input interface. If no appropriate implementation is found, the developer can create her own implementation, which can afterwards also be stored in the component pool. Also, if no appropriate component is found, the developer can implement her own component, which can then also be stored in the pool. This resource sharing mechanism provided by the component pool is one of the fundamental ideas behind the IR-BASE proposal.
The IR-BASE component pool is currently under development. When ready, it should be available as a web service.

## 4. APPLICATION OF IR-BASE TO THE TEACHING OF INFORMATION RETRIEVAL

IR-BASE was designed with the general purpose of allowing the rapid development and experimentation of IR solutions, mainly for the research and teaching of Information Retrieval. This goal is achieved by providing a framework for the creation of interoperable system components. IR systems developed under IR-BASE follow a strict component-based architecture, were we can view each component as a black-box, communicating with other components only through a set of predefined methods. This architecture is ideal for use in teaching and learning IR technologies, for several reasons.
First, it allows students to study an IR system at different levels. Viewing a system as a set of independent interacting components will allow a high-level illustration of how the system works and how it can be configured and applied to different problems. Students can disassemble and reassemble it using alternative implementations of the same components, they can study its overall behavior and try it on different settings. Since each component is, in itself, an independent system, students will also be able to study each one individually, at a lower level. They can reimplement components using different techniques and easily insert them into a working IR system.
Second, the use of probing interfaces will allow the visualization a component’s internal functioning. This is mostly useful for the illustration of IR algorithms and techniques. For instance, students will be able to ‘‘see” an inverted file being built by an indexer component, or document groups being formed by a clustering component.
Third, by providing a standard workbench, the students’ work can be shared and reused. Components developed for course projects can later be used, studied, and applied by students in following years, or even in different institutions.
With this in mind, the NAÏF (**N**early **A**ppropriate for **I**R **F**unctionality) system is currently being developed as part of the IR-BASE framework. NAÏF is a simple IR system, capable of collecting web pages, indexing their textual content and providing a search mechanism. NAÏF will be completely implemented using the IR-BASE framework. On a first approach, the NAÏF system will contain a web crawler component, a document parser component, a text indexer component, and a document ranking component. Its main purpose is to serve as a usable demonstration of a simple IR system and to allow both students and researchers to study and modify it freely.

## 5. CONCLUSIONS AND FUTURE WORK

The IR-BASE framework proposes the use of a component-based architecture, together with an organized repository of components, as a means to rapidly develop IR systems. The emphasis of IR-BASE is on modularity and interoperability. By ensuring a set of simple guidelines on how to build components, it can provide a way of integrating code from different sources which might have been developed for completely different goals.
IR-BASE is of interest to IR researchers who want to develop experimental solutions that are easily configurable, modifiable and extensible, to teachers who want to provide their students with a basic IR framework for course projects, and to IR professionals who want to rapidly develop prototypes to test new ideas. We expect IR-BASE to be a major contribution to the integration of teaching and research efforts and to the cooperation between research groups and institutions, by providing an effective way of exchanging and sharing implementations and documentation of different IR solutions.
The project is currently under development. The framework has been completely specified and is now being implemented. The next steps in this process will be the creation of the IR-BASE component pool, including the component storage and search services, and the implementation the example IR system, NAÏF.

## ACKNOWLEDGMENTS

This work was financed by FCT project ``An Integrated Framework for the Research and Teaching of Technologies for Information Retrieval", ref. POSC/EIA/58194/2004.

## REFERENCES

[1] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley-Longman, 1999.

[2] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, pages 35-43, May 2001.

[3] Chris Buckley, Gerard Salton, James Allan, and Amit Singhal. Automatic query expansion using SMART. In *Third Text REtrieval Conference (TREC-3)*, pages 69-98, November 1994

[4] W. Buntine, K. Aberer, I. Podnar, and M. Rajman. Opportunities from open source search. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2005)*, Compiègne, France, September 2005.

[5] The Lemur toolkit for language modeling and information retrieval. http://www.lemurproject.org/.

[6] Apache Lucene. http://lucene.apache.org/.

[7] Iadh Ounis, Gianni Amati, Vassilis Plachouras, Ben He, Craig Macdonald, and Douglas Johnson. Terrier information retrieval platform. In *Proceedings of the 27th European Conference on Information Retrieval (ECIR 05)*, Santiago de Compostela, Spain, March 2005.

[8] Fei Song and W. Bruce Croft. A general language model for information retrieval. In *Proceedings of the 8th International Conference on Information and Knowledge Management (CIKM 1999)*, pages 316-321, Kansas City, MO, USA, November 1999.

[9] WIRE - Web Information Retrieval Environment. http://www.cwr.cl/projects/WIRE/.

[10] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishers, 2nd edition, 1999.