Electronic Publishing, Storage, Dissemination and Retrieval of a Scientific Journal through the Web

M. Agosti  F. Crestani  M. Melucci
Dipartimento di Elettronica e Informatica  Department of Computing Science  Dipartimento di Elettronica e Informatica
Università di Padova  University of Glasgow  Università di Padova
Via Gradenigo 6/A,  Glasgow G12 8QQ  Via Gradenigo 6/A
I-35131 Padova, Italy  Scotland  I-35131 Padova, Italy
agosti@dei.unipd.it  fabio@dcs.gla.ac.uk  melo@dei.unipd.it

Abstract
The initial part of the paper describes the general architecture of a prototype system developed in the context of the IRIDES project for the electronic publishing, storage, dissemination and retrieval of The Computer Journal, published by Oxford University Press. The core of the paper is devoted to one of the components of the overall system: AUCTOR. AUCTOR is a system for the automatic authoring of a HTML hypertext representing the content of journal's articles. After a detailed description of the system, considerations on the automatically authored hypertext are given. AUCTOR in particular, and IRIDES in general, can be considered as the initial component of a complete environment to collect different collections of documents and services for the implementation of a modern digital library.

1 Introduction
The work reported in this paper is in large part based on research results reached by the Information Management Systems (IMS) group of the Department of Electronics and Informatics (DEI) of the University of Padua (Italy) in the context of the IRIDES project. The IRIDES project has been focused on the design and development of a prototype for the electronic publishing, storage, distribution and retrieval of a scientific journal over the Internet. The Computer Journal, published by the Oxford University Press (OUP), has been used for the design and development of a working prototype that has served as a validation tool of the adopted design methodology. The Computer Journal was chosen because it can be used as a significant example of a widely distributed scientific journal.

The paper is organised as follows: Section 2 sets the context of IRIDES and explains why it is the kernel of a complete environment for a digital library. Section 3 gives a general overview of the IRIDES project. Section 4 presents relevant results of the work carried out at Padua in the context of IRIDES. Section 5 describes briefly the evaluation of the work and gives some final considerations for the evaluation of the work and reached results. Section 6 gives in general directions for further work in the area of automatic authoring of hypertext, and specifically for future work derived from IRIDES results.

2 Digital Libraries and the IRIDES Project
The IRIDES project was carried out in 1995/96 by a joint force of researchers from three different research institutions: the Computing Science Department of the University of Glasgow, the Institute for Integrated Publication and Information Systems (IPSI) of the GMD Darmstadt, and the IMS group of the Department of Electronics and Informatics of the University of Padua.

The aim of the project was to develop a prototype for the electronic publishing, distribution and retrieval of a scientific journal over the Internet. The theoretical work was general with the scope of de-
signing a methodology that would suit any scientific journal. However, the prototype development focused on a particular case, the electronic publishing, distribution and retrieval of The Computer Journal of the Oxford University Press. The Computer Journal is a widely distributed scientific journal, published in UK but distributed world-wide for over 40 years, providing a complete overview of developments in the field of Computer Science.

The documents to be considered in the project were the digitised forms of the articles of the journal that traditionally are printed on paper. Accordingly with a wide interpretation of the concept of a Digital Library (DL) [14], that says that a digital library is not merely equivalent to a digitised collection with information management tools, but it is rather an environment to bring together different collections and services in support of the full documents life cycle, we have considered the design of the IRIDES project environment as an open one. IRIDES is the kernel for an open environment where it is possible to insert or to make possible the co-operation among different services and collections. The initial documents collection consists of scientific articles, but, after the validation of the the project, it will be possible to add further services for the final users.

The tasks addressed by IRIDES have been mostly concerned with the design and implementation of information management tools, but the tools were based on different information management and querying paradigms, to show that in a digital library environment it is possible, if not mandatory, to implement services that support different ways of searching and finding information. Final users of a DL have different searching capabilities together with different information requirements. The services rendered by IRIDES can be seconded by the implementation of complementary services to fully implement a modern digital library.

3 The IRIDES Project

This paper presents in detail aspects related to the only one component of the IRIDES project. However, to give to the reader a general overview of the overall system, without entering into the details of the work carried out, we report here a list of the tasks faced in the context of the project to give the reader an overall idea of the complete project that is completely described in [10]:

1. Identify the target user population and the user requirements. Identify a smaller group of users inside the members of IDOMENEUS on which the testing and evaluation of the prototype will be done.

2. Identify and study the requirements of the publisher, in particular of the Oxford University Press.

3. Evaluate the costs and benefits of providing an electronic version of a scientific journal in comparison to its classical paper form.

4. Study existing document mark-up languages and choose the most appropriate for the development of a prototype for the electronic publishing and authoring of a scientific journal.

5. Develop a technique for the automatic authoring of the journal’s content. The technique must take into account that the journal is multimedia.

6. Identify and study existing tools and standards for information dissemination over the Internet. Analyse the extent to which these existing products and facilities can be used for the electronic distribution of a scientific journal.

7. Develop or modify an existing interface for information dissemination to match the requirements of the documents to be represented.

8. Identify the design and storage issues of a large hypermedia repository of scientific articles. This involves the problem of storing highly structured multimedia objects, as well as integrating database and information retrieval techniques to provide adequate access based on structure and contents of the database.

9. Develop tools to aid the user to find journal articles relevant to his information need. This will work both inside the current issue of the journal and in an on-line repository of past issues. Use state-of-the-art information retrieval techniques to enable the user to express his information need using natural language.

10. Develop tools to aid the user to browse the journal content in a non-linear way, enabling him to discover information relevant to his information need by serendipity.

11. Study techniques that enables the publisher to guarantee intellectual property rights and to charge for the on-line access of the journal.
12. Develop tools to enable the publisher to deliver by electronic mail a hard copy (postscript) versions of articles. The tools should guarantee intellectual property rights and enable charging the user for the service.

These tasks were tackled by the three research groups in an independent, although co-ordinated way, for developing a system composed of three major components:

- the information searching and retrieval component;
- the automatic authoring component;
- the storage and dissemination component.

A schematic view of the overall architecture of the prototype system developed in the context of this project is depicted in Figure 1.

A detailed description of all the components is outside the scope of this paper, but a brief description of how they interact with each other is here introduced.

The information searching and retrieval component is centred on a probabilistic IR system developed at the University of Glasgow [13]. The IR system enables the user to search the content of the articles of the journal so that he can find the article(s) that most suits his information need.

The automatic authoring component is centred on the AUCTOR system, a fully automatic authoring system that transforms a journal article written in SGML format into a set of HTML pages that can be made available through the World Wide Web (WWW or Web). The automatic authoring system is based on a methodology [5] and on an initial prototype [2] designed and developed at the University of Padua (Italy).

The storage and dissemination component is centred on a distributed object-oriented multimedia database management system developed at GMD-IPSI, Darmstadt (Germany) called VODAK [1]. VODAK is a very complex tool enabling the efficient storing and management of multimedia objects. In particular VODAK deals with the storage of the journal articles and with all the operations related to the handling the user’s purchases.

The above three components interact with each other in two different “usage cycles”: a publisher cycle and a user one. In the following two sections we will explain these two usage cycles to give an idea of how the components interact with each other.

3.1 The Publisher Cycle

The publisher cycle is the sequence of operations concerned with making available to the user through different channels an electronic version of the journal. This production line is depicted with a dotted line in Figure 1.

The input of this cycle is an SGML version of the articles of an issue of the journal. These are fed into the storage and dissemination component that produces a PDF version of each paper. This version is kept in store for future delivery to interested users, and an ASCII version of each paper, that is fed to both the automatic authoring component and the information searching and retrieval component. The automatic authoring component takes the ASCII version of the paper and produces a hypertext version of part of the paper (title, author(s) name(s), and abstract) in HTML that is made available for browsing on a Web site. The information searching and retrieval component receives the ASCII version of the paper and indexes it using articles sections as the basic searching and retrieval structure.

3.2 The User Cycle

The user cycle is the sequence of operations a user is expected to perform in order to find, browse, order, and receive one or more articles of the journal. All these operations are done on-line via a Web browser, like for example Netscape or Microsoft Explorer. The user cycle is depicted with a continuous line in Figure 1.

In the user cycle the user connects to the service using a Web browser. At this stage he does not need to have an account with the service. He can freely search and browse the journal repository. The first operation the user will probably perform is searching the content of the repository for articles that are of his interest. If the user knows already the full details of the article he is looking for, he can bypass this operation. The information searching and retrieval component, upon receipt of a user’s natural language request through the Web interface, produces a ranking of sections extracted form articles estimated to be relevant to the user need expressed in the request. The purpose of displaying article sections and not the full text of the article or simple a bibliographic reference is twofold. It enables the user to assess the relevance of the articles founds and, at the same time, it does not disclose the full content of the articles without the publisher’s clearance and/or charging the user.

The user look through the query results and select one of the sections as relevant to his needs. This brings him into the hypertext that the automatic authoring
Figure 1: The architecture of the IRIDES prototype
component has produced. By selecting a section of interest the user is taken to the full bibliographic details of the paper from which that section come from. From here the user can browse the entire collection of articles in current and past issues of the journal, searching for other articles that are related to the one he just selected. The browsing of the hypertext structure can be achieved in different ways using the complex networks of relationships built by the automatic authoring component. We will discuss this in more detail in the remainder of the paper.

Once the user is happy with his search and browse results and decide to order some of the articles he found to be relevant, he can do so on-line using the same Web browser. The user place an order for some articles, which is processed by the storage and dissemination component. If the user has an open account with the system, then he is asked by the storage and dissemination component to provide personal details for the charging for the service and for the delivery of the purchase orders.

The user cycle can be repeated as many times the user requires and halted at any desired moment.

The system can store the information about the user interests that is implicitly provided by the search and browsing of the repository. This information can be used to inform the user if a new article related to his information interests is published in future.

4 Electronic Authoring of a Scientific Journal

To make the electronic authoring of a scientific journal effective we need three elements:

1. a hypertext model which help us describe and capture the semantics of the collection of journal abstracts,

2. a methodology describing the necessary steps to transform journal articles available in machine readable format into hypertext,

3. a automatic authoring component implementing the methodology.

This paper concentrates on the design and implementation of the fully automatic authoring component, called AUCTOR. By “fully automatic” we mean that the process is performed without any human intervention. The importance of having a fully automatic authoring component within a system such as the IRIDES one has many motivations:

1. it enables fast re-production and publishing of large quantities of electronic material to be used for searching purposes;

2. it offers alternative information searching functionalities, such as browsing, to be integrated to the most classical ones, i.e. free-text and field-based searching;

3. it is faster and uses less human resources than a manual authoring procedure.

4.1 The Hypertext Model

A hypertext conceptual model has been adopted to describe the abstract collection semantics [4]. The model is depicted in Figure 2.

![Figure 2: The conceptual hypertext model](image-url)
thesaurus of the application domain the collection belong to), and are related through semantic pre-defined relationships. The basic difference between the set of index terms and the conceptual network is that the former is a function of the document collection, and the latter is independent on any collection of documents. The role of the set of index terms is to provide an interface between the language used by the authors for writing the documents, and the language of the conceptual network.

The above conceptual model is implemented into system architecture as depicted in Figure 3.

![Figure 3: The hypertext architecture](image)

This is the final output of the authoring process. Each white box corresponds to a level of the conceptual model. The first level (D) is that of the journal articles or documents, the second level (T) contains the index terms, and the third level (C) contains the concepts. Conceptual links between nodes are implemented through a list of links depicted as grey boxes. For example, TD links are stored within a list placed between a T node and D node. As nodes are conceptually related through many-to-many relationships, lists store different links, i.e., one link for each destination node. The way these lists are constructed depends on the specific techniques employed. The following Section illustrates how IR techniques provide with methods to perform the construction of such lists.

### 4.2 The Methodology

The methodology describes a process for the automatic construction of a hypertext by transforming a “flat” collection of document texts into a hypertext. The basic peculiarities of the proposed methodology are:

- The process of automatic authoring takes as input a set of full-text documents without requiring any pre-defined or manual inserted link or mark-up language tag. This means that the methodology can be generally used for any text collection.

- Sound statistical and probabilistic IR techniques are employed to construct links of different types. In this process, in fact, we prefer take advantage of well-known IR techniques and to combine them into a new methodology rather than propose brand new techniques that need to be tested. In order to make the whole hypertext construction process automatic, the methodology is heavily based on the automatic indexing process for long in use in IR (see [12] for example).

- The hypertextual network is made explicit in order to make clear to the final user the way the system represents the content of documents. Links and nodes are implemented through the HTML constructs, and we have therefore sacrificed the quality of the user-computer interface in order to make hypertexts as widely usable as possible.

The aim of the methodology is to automatically construct the nodes and the lists of links in two main automatic steps: indexing and authoring.

**Indexing** This is the first step to be performed before any others since all links and most nodes are built on the basis of indexing data. It starts from the “raw” data, and the result is the creation of a set of index terms nodes linked to the documents. We used the vector-space model [12] where a document \( n \) is described as a vector \( \vec{n} = (n_1, \ldots, n_K) \), where \( K \) is the total number of index terms, and \( n_k = 1 \) if term \( k \) occurs in \( n \), 0 otherwise. We do also represent terms in a similar way. Term \( k \) is described as \( \vec{k} = (k_1, \ldots, k_N) \), where \( k_i = 1 \) if document \( i \) is described by index term \( k \), 0 otherwise. One can compute term weights \( w(k) \) for each index term \( k \).

**Authoring** This step is performed using the data resulted from the indexing process. The result is the construction of the lists of links in two main automatic steps: indexing and authoring.
by AUCTOR as effective for retrieval purposes. A threshold value can be used to cut off the least useful links in the list. IR provide useful threshold functions, either based on probabilistic models [15, 8] or on the vector-space model [12].

The algorithms to be used for the construction of the lists vary accordingly to the specific type of list. In particular, DT and TD list scores are based on the $f_{ik} \times w(k)$ provided $f_{ik}$ the frequency of occurrence of term $k$ within document $i$. As the $w(k)$ weight is constant over all documents, TD links are ranked to the document term frequency $f_{ik}$. The construction of similarity links, i.e. DD and TT is based on similarity functions defined on the document and index term descriptions. If one consider the vectors representing documents and index terms, the similarity function between two information objects is based on the angle between the corresponding vectors [12]. In a similar way, CT links are ranked to the $w(k)$ weight, i.e. the higher the $w(k)$ weight, the higher the rank of $k$ index term within the list of links associated to a concept. TC links are ranked according to a slightly more complex algorithm. Concepts described by a given index term $h$ are identified. For each concept $c_i$, its vector-based description $\tilde{c} = (c_1, \ldots, c_K)$, such that $c_k = 1$ is considered. Then concepts are ranked to the corresponding sum of index term weights $\sum_{k=1, k \neq h}^K w(k)$. Thus, concepts described by a high number of “important” index terms are ranked at the top of the list, where by “important” we mean “with a high term weight”. TC links represent the semantic relationships occurring within a classification scheme or thesaurus. The corresponding links are then manually and very carefully inserted. Their effectiveness is due to the fact that they have been decided by a pool of expert users, and therefore they can be presented to the user without any automatic authoring algorithm.

4.3 The AUCTOR System

AUCTOR makes the integration of searching and browsing facilities effective, since it implements the methodology for the automatic construction of a hypertext made of journal articles or abstracts, the index terms of the collection, and the ACM classification system.

The input data are:

- a “flat” collection of “flat” texts in HTML format,
- a list of stop words, also called “stop list”, including the words considered to be common, and then non-significant to the collection domain,
- a text storing a classification scheme to be used as a tool implementing the network of concepts. The current implementation of AUCTOR employs the classification system$^2$.

The output produced by AUCTOR is a set of HTML files stored as a directory hierarchy and managed by a file system. The HTML files can be accessed through a Web server which communicate with any Web client, such as Netscape.

AUCTOR is made of two components (see Figure 4):

Figure 4: The IR Object Builder and the Hypertext Author

IR Object Builder (IROB) takes as input the text collection and the stop list, and produces the first two levels of the hypertext, i.e. document nodes, and index term nodes. Stop word removal, stemming, and weighting are described in [12].

Hypertext Author (HTA) performs the second and third steps of the methodology. HTA takes as input what IROB produces, as well as the ACM classification scheme, generates the third hypertext level, and then creates all the links between the nodes of the hypertext.

At the beginning of Section 4, we stressed the importance of a component, such as AUCTOR in the IRIDES system. More generally, AUCTOR could be inserted in other systems that publish electronic material. Figure 5 depicts a simplified version of the electronic publishing process as described in [6] where a new type of information producers, called “secondary information producers”, such as indexers, abstractors, or converters, are inserted in the publishing process.

They interface the publisher and the users by providing with new services of information retrieval and dissemination. Within such a process, AUCTOR receives the electronic text from the author or the publisher, which can provide AUCTOR with the HTML version of the text. AUCTOR automatically builds a hypertext which nodes are journal abstracts and links are based on node content. The final hypertext can be accessed through a Web server which communicates with any Web client, such as Netscape. The communication between servers enables the delivering on demand of the relevant articles. AUCTOR thus works as a new type of secondary information producer since it provides with the new service of information retrieval and dissemination.

Figure 5: The publishing process and the role of AUCTOR

5 Evaluation

The main objectives of the evaluation of IRIDES work were (1) to test the acceptance of the IRIDES system in the scientific user community, and in particular in the community of the Computer Journal readers to which the system itself is applied, and (2) to collect suggestions about the directions to take to improve the current prototype.

Specifically, the evaluation phase aimed at testing if the hypertexts built by AUCTOR are effective for the searching and retrieval of information from the Computer Journal articles. More generally, it was believed that the evaluation results could give useful insights to assess if this type of hypertexts could be employed as a complementary service to the other different ways of searching and retrieving information from digital library.

The evaluation was based on the empirical observation of the system utilisation by a pool of users. We chose to select experts and scientists in the area of information retrieval and hypermedia. The experts were in the role of potential end-users and were asked to solve some search tasks. Scientists were a kind of “reviewers” which gave their opinions about the usability and the faultiness of the system. This choice permitted us to collect very useful and precise suggestions to improve the prototype. Other evaluation measures, such as precision and recall were considered insufficient since the aim of evaluation was to test the overall system performance and usability, rather that the retrieval effectiveness. The users were asked to “think aloud” during the interaction with the prototype. Comments and interviews were recorded on paper.

Without entering into the details of the evaluation of the IRIDES project as a whole, the final evaluation of the prototype brought the following findings for AUCTOR:

- A hypertext automatically produced by AUCTOR seems to respond to the users’ needs, since users spontaneously used it whenever the search task could be effectively carried out through browsing. In particular, the ACM classification scheme helped users searching relevant articles.

- Once a concept of the ACM classification scheme is reached, the users expect to find a cluster of linked documents. The hypertext architecture does on the contrary propose to the user a cluster of automatically extracted index terms, and this design choice disoriented the users. We have taken into consideration this finding and in [9] we have modified the hypertext architecture and the automatic authoring algorithm accordingly.

- The users would like to combine browsing and searching capabilities; they asked for the possibility of selecting index terms or concepts of the ACM classification scheme and building a query during the navigation. We are currently working towards adding a retrieval engine (more complex that simple string searching already provided by NetScape) that will enable the user to build queries by picking up index terms or concepts during the browsing. Some first steps towards this end have been presented in [3] and [7].

- The number of articles stored in the database and the indexing algorithm are critical issues for the effectiveness of the statistical and probabilistic techniques underlying the automatic authoring algorithm. For example, the list of index terms may be disorienting or annoying if either the stemming algorithm is too trivial or the quantity of data is too small.
• During the design of the hypertext we chose to present the list of links ranked by a score, but to avoid the displaying of the score itself since we assumed that the user would not have been interested to know it. On the contrary, experiments suggested that ranking is useful and displaying the score makes the ranking criterion clearer.

• At first glance users are pleased by the presence of similarity-based links between abstracts and index terms; however, after having examined similar nodes, they asked for the meaning of the similarity measure. This suggests that we have to provide an online help that explains this feature.

• The users would like to have a more detailed content description of the semantics of links and nodes (in particular of index terms nodes). The description currently provided by the prototype is not detailed enough.

Taking into consideration the results of the evaluation of AUCTOR in the context of IRIDES, the following general considerations on the usability of hypertext for information searching and retrieval arise.

Multi-level hypertexts, such as the ones automatically produced by AUCTOR provide with a conceptual framework which reduces the risk of disorientation through a better communication with the system than with a traditional information retrieval system. The better effectiveness can be obtained thanks to the perception the user can have of the different abstraction levels by which node types are organised. The problems arised by the users and referred previously in the last two points may be due to the lack of powerful hypertext capabilities of the Web user interface we chose for the IRIDES system. In particular we would like to stress the lack of possibilities in rendering the different node types of the hypertext on the screen at the same time.

The difficulty to find a “good” structure for the Web pages of the hypertext has been the main consequence of the lack of graphical capabilities. The main issue is whether to present to the user a number of small pages, or alternatively few and large pages. The former option has the advantage of keeping the architecture of the model of reference (Figure 3), but the user should then click many pages that only include anchors, i.e. the pages connecting the levels. The latter option reduces the number of pages to be browsed before getting relevant abstracts by embedding more links in one page, but it has the disadvantage to be too large: for example, the index term page might include a lot of links to objects placed on the three levels. We chose to implement few and larger pages because this requires less time for browsing. This is important especially within a distributed system such as the Web, since host connections and document receiving require seconds, sometimes minutes.

6 Future Work

After the completion of the IRIDES project, the IMS group has reconsidered the functionalities available in AUCTOR, and the lack of the incremental updating functionality has been regarded as a shortcoming whenever the authoring tool needs to be used in the context of more complex applications such as those of interest for digital library implementations [11].

A proposal for a new methodology able to manage and retrieve information stored in large collections by means of “on-the-fly” automatically authored hypertexts is under study and an initial proposal is given in [3]. Further work is necessary for the development of an operative efficient prototype to be scalable to operative implementations, but the initial results are encouraging.

Acknowledgements

The design and development of the prototype system presented in this paper was supported by IDOMENEUS, the ESPRIT Network of Excellence No. 6606. IRIDES has been a joint project together with Glasgow University and GMD Darmstadt research teams. We acknowledge the collaboration of those teams, that have designed and developed the other parts of the IRIDES prototype that are complementary to those here presented.

References


