Chapter 1

Introduction

The World-Wide Web contains hundreds of millions of pages of data. Existing information retrieval techniques are inadequate for navigating this unprecedented cornucopia of semistructured information. Turning the Web into "the world's largest knowledge-base" has been proposed as one of the most important challenges in artificial intelligence [27]. This is the problem that I address.

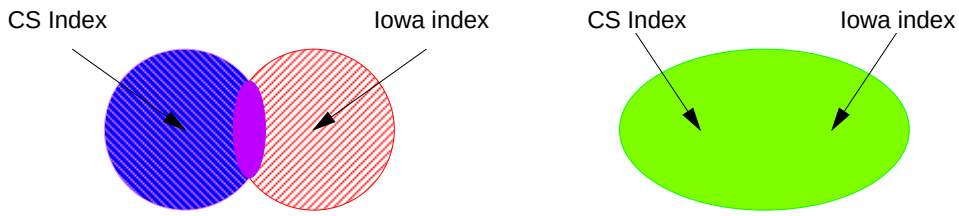
The success of the Web is a testament to the importance of *structure*. Users of the Web make use of both intra-document structure (via annotations indicating headers, lists, and formatting directives) and inter-document structure (hyperlinks). Even the uniform resource locator (URL) identifying a page is structured: typically a host name followed by a location in the file hierarchy. All of these types of information are used by human readers but largely ignored by automated Web tools. I argue that this information is potentially valuable to automated tools and show the feasibility of making use of structural information through a system that allows easy creation of such applications.

While information retrieval has been studied for decades, new techniques are needed because the domains examined have been very different from the Web, both quantitatively and qualitatively. The most studied domain is "flat" text collections, i.e., sets of articles or documents, each of which is a sequence of unannotated text. Most current Web tools are based on information retrieval techniques developed for flat text collections, ignoring the Web's structural information. Similarly, the Web is different from "classical" hypertext applications, such as an encyclopedia available on CD-ROM, which is static and authored by a single team, as will be discussed in more detail in the next chapter. Because of the novelty of the Web, new information retrieval paradigms are needed.

The solution I present is viewing the Web as a giant database, where structural relations are represented as database relations, which provides a uniform framework for accessing the diverse types of Web structure. This allows the construction of powerful "ParaSites", which mine and apply information available on the Web in ways unanticipated by the information's authors. Specifically, this dissertation provides the following:

1. an ontology specifying types of structural information to extract from the Web and a corresponding relational database schema
2. a set of algorithms for representing an unbounded set of information in a finite database
3. a system, Squeal, that implements the abstraction of the Web as database
4. a set of ParaSite applications, to show the value of tapping this underutilized structural information

In this introduction, I motivate the work by describing some ParaSites that can be easily built on top of Squeal.



(a) Intersecting circles **(b) An ellipse**

Figure 1-1: A geometric representation of material related to computer science and Iowa

1.1 ParaSites

Human beings, not machines, are the ultimate authorities on intelligent information retrieval. Machines just have the advantage of being able to logically process data more quickly than human beings can. The best approach to information retrieval (or any problem) is to assign to the computer and the human being the parts of the task that each is best at. Specifically, the computer should execute an algorithm that reduces a new task to a set of problems previously solved and documented by human beings. In other words, the computer should act as a parasite (ParaSite). Hence, my approach is to provide a system that breaks a task down into a set of queries on stored human knowledge (the Web) and puts them back together to produce an answer. The rest of this section describes some useful ParaSites that show the value of being able to access the Web's structural information.

1.1.1 Link geometry

Consider the set of pages within three hyperlinks of a computer science index. The pages within one link of the index are almost certainly related to computer science; the pages two or three links out have a lower probability, although still greater than that of random pages on the web. We can think of the set of pages within n links of an index I as being the interior of a circle (hypersphere) of radius n with center I. We could create two such circles with centers (foci) representing different subject indices and intersect them in an attempt to find material that is relevant to both subjects, as shown in Figure 1-1a, or we could build an ellipse from the two foci, as shown in Figure 1-1b. Indeed, crawls from Yahoo's Computer Science (CS) and Iowa indices meet, appropriately, at Grinnell (Iowa) College's Department of Mathematics and Computer Science (www.math.grin.edu), showing that link geometry can be a useful technique for finding pages on a given set of topics. This technique applies a very simple algorithm to human-created information on the Web to create a new and useful application.

1.1.2 A recommender system

One useful class of information retrieval applications is *recommender systems* [33], where a program recommends new Web pages (or some other resource) judged likely to be of interest to a user, based on the user's initial set of liked pages P. A standard technique for recommender systems is extracting keywords that appear on the initial pages and returning pages that contain these keywords. Note that this technique is based purely on the text of a page, independent of any inter- or intra-document structure.

Another technique for making recommendations is collaborative filtering [40], where pages are recommended that were liked by other people who liked P. This is based on the assumption that items thought valuable/similar by one user are likely to by another user. As collaborative filtering is currently practiced, users *explicitly* rate pages to indicate their recommendations. We can think of the act of creating hyperlinks to a page as being an *implicit* recommendation. In other words, if a person links to pages Q and R, we can guess that people who like Q may like R, especially if the links to Q and R appear near each other on the referencing page (such as within the same list). This makes use of intra-document structural information.

Accordingly, if a user requests a page similar to a set of pages $P_1, \ldots, P_n$, a ParaSite can find (through AltaVista) pages R that point to a maximal subset of these pages and then return to the user what other pages are referenced by R. Note that ParaSite does not have to understand what the pages have in common. It just needs to find a list that includes the pages and can infer that whatever trait they have in common is also exemplified by other pages they point to.

For example, the first page returned from AltaVista that pointed to both Computer Professionals for Social Responsibility ("www.cpsr.org/home.html") and Electronic Privacy Information Center ("www.epic.org") was a list of organizations fighting the Communications Decency Act; links included the Electronic Frontier Foundation ("www.eff.org") and other related organizations. Recommender systems will be discussed in greater detail throughout the thesis, where it will be shown that the ParaSite approach to this problem is very effective.

1.1.3 A home page finder

A new type of application made necessary by the Web is a tool to find users' personal home pages, given their name and perhaps an affiliation. Like many information classification tasks, determining whether a given page is a specific person's home page is an easier problem for a person to solve than for a computer. Consequently, ParaSite's primary strategy is not determining directly if a page "looks like" a home page but finding pages that *human beings* have labeled as being someone's home page. While there is no single stereotypical title for home pages, there is for the anchor text of hyperlinks to them: the author's name. For example, an AltaVista search for pages containing the name "Nicholas Kushmerick" returned 27 links, none of them to his home page. In contrast, a search for hyperlinks with *anchor text* "Nicholas Kushmerick" returned three matches, two of which were links to the correct home page and one to his email address. This is an example of taking advantage of inter-document structure.

Another class of useful structural information is intra-document structure. For example, if the name "Nicholas Kushmerick" appears in the title field of a page, that is more significant than if it appears in the body. The structure of the URL can also be used. The URL of Kushmerick's home page is: "http://www.cs.washington.edu/homes/nick/". This is easily recognized as a likely home page because:

1. The file name is the empty string. (Other stereotypical file names for home pages are "index.html" and "home.html".)
2. The final directory name is the user's email alias.
3. The penultimate directory name is "homes". (Another common penultimate directory for home pages is "people".)

This application will also be discussed at greater length later in this thesis.

1.1.4 Summary

These applications all make use of structural information on the Web. Squeal is the first system to allow their easy implementation. Furthermore, Squeal provides a clean interface that treats the different types of structure in a uniform manner, allowing the user to ignore the details of how the information is represented and to focus on the more meaningful aspects.

source	anchor	destination
www.ai.mit.edu	Lab for Computer Science	www.lcs.mit.edu
www.ai.mit.edu	MIT	web.mit.edu
www.ai.mit.edu	People	www.ai.mit.edu/people
www.lcs.mit.edu	AI Lab	www.ai.mit.edu

Table 1.1: Sample Relation **link** Representing Hyperlinks. Each row or "line" contains a **source** field, an **anchor** field, and a **destination** field. The first line represents the hyperlink labeled "Lab for Computer Science" from the page with URL "www.ai.mit.edu" to the page "www.lcs.mit.edu".

1.2 A Database Interface to the Web

Relational databases and Structured Query Language (SQL) were designed to provide users with a powerful way of accessing structured information. Squeal makes use of these technologies in both its user interface and its implementation. Specifically, it provides the user with the illusion that the Web is stored in a well-defined relational database and can thus be accessed through SQL. The Squeal implementation also stores information, although not the entire Web, in a relational database.

1.2.1 Background

A *relational database* [6] is used to represent sets of data. The objects and constraints of a database are defined by its *schema*. The only complex data structure is the *table*, which consists of zero or more *rows*, each of which is a set of one or more *columns*, each of which can hold data of a specific type, such as an integer or a character array. Table 1.1 shows a table named **link**, each of whose rows consists of a set of three columns, named "source", "anchor", and "destination". Such a database might be used to store information about hyperlinks among web pages, as illustrated with the sample data.

Structured Query Language (SQL) [7] is a declarative language for accessing the schema and data of a relational database. The command for retrieving data is SELECT with a required FROM clause and an optional WHERE clause. The FROM clause indicates what tables should be queried, and the WHERE clause constrains the cases for which data is returned. Some sample SELECT statements and the resulting tables follow:

What are the destinations of hyperlinks originating at "www.ai.mit.edu"?
SELECT destination
FROM link
WHERE source = 'www.ai.mit.edu'

destination
www.lcs.mit.edu
web.mit.edu
www.ai.mit.edu/people
www.ai.mit.edu

Show the values of all fields on lines with anchor text "MIT".
SELECT *
FROM link
WHERE anchor = 'MIT'

source	anchor	destination
www.ai.mit.edu	MIT	web.mit.edu

Figure 1-2: Cache SQL database to the Web

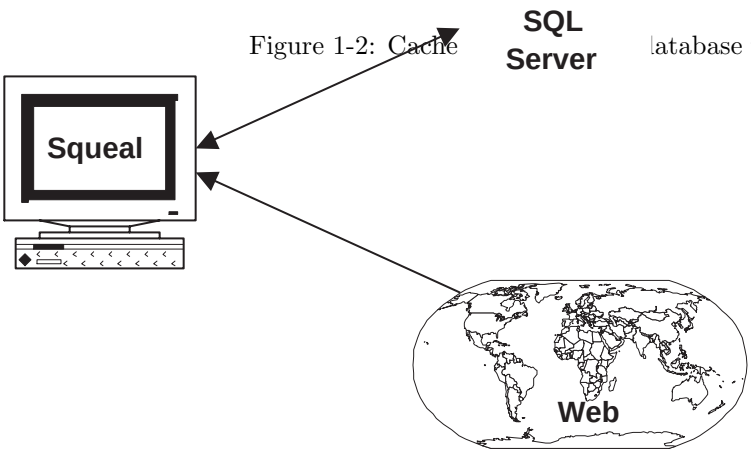


Figure 1-3: Structure of Data Transfer in the ParaSite System

What pages point to each other?
SELECT l1.source, l2.source
FROM link l1, link l2
WHERE l1.source = l2.destination AND
l2.source = l1.destination

l1.source	l2.source
www.ai.mit.edu	www.lcs.mit.edu
www.lcs.mit.edu	www.ai.mit.edu

It would be very useful for tools and users to be able to access the Web in this fashion.

1.2.2 Squeal

Despite the illusion presented by the **link** table, the Web is not really represented as a relational database. Instead, it is a collection of documents stored on many different machines. My system, Squeal, bridges the gap by retrieving and processing information from the Web as needed in order to maintain the illusion that the Web is a relational database. As Figure 1-2 illustrates, this is done by having an actual relational database act as a cache of the Web.

The structure of system data flow is shown in Figure 1-3. The user enters commands in Squeal on a client machine. The Squeal interpreter fetches the appropriate pages from the Web, parses them, places them in an underlying SQL database, and queries the database in SQL to answer user queries. This provides the user with the ability to query the Web as though it were a relational database.

Squeal's core is syntactically a subset of SQL, although interpreted in a different way. Unlike ordinary SQL interpreters, querying the database with the Squeal interpreter can cause the state to change. For example, consider the SQL command:

SELECT * FROM link WHERE source = 'www.ai.mit.edu'

In an ordinary SQL implementation, this would retrieve all lines of the **link** table whose **source** column is equal to "www.ai.mit.edu". The Squeal interpretation of the same statement is:

1. Check whether the system has already computed the links from the page "www.ai.mit.edu". If so, go to step 3.
2. Retrieve the page and parse it, putting the appropriate entries in the **link** table (possibly modifying other tables).
3. Do the ordinary SQL interpretation of the original command.

Except for a difference in response time, the user cannot tell whether the information was already in the database or if it had to be retrieved, processed, and inserted into the database. In other words, the database acts like a cache, as illustrated in Figure 1-2. The result is the illusion that the user is directly querying the Web. Once this abstraction is in place, the discussed ParaSites can be easily implemented.

1.3 Related work

Related work that serves as background is discussed in this section. A comparison of Squeal to other database interfaces to the Web appears in the conclusion, after Squeal has been more fully described.

1.3.1 Semantic networks

The precursors of this work hearken all the way back to the nineteen-sixties. Treating the Web as a knowledge base is reminiscent of semantic networks [31, 26], graphs whose nodes represent concepts or entities and whose arcs represent the relations among them. In the seventies and eighties, Woods [46] and Trigg [43] improved on the model by distinguishing among different types of relations between nodes, also applicable to the Web.

1.3.2 Structure within documents

Numerous hypertext researchers, including Furuta [12] and André et al. [3] have written about structured documents and the benefits they provide for consistency, reusability, and verifiability. Wilkinson and Fuller discuss the use of intra-document structure for both answering queries and displaying results [45]. Before the rise of HTML, the markup language most commonly considered was Standard General Markup Language (SGML), leading to the development of HyQ [18], a Lisp-like query language for the Hypermedia/Time-based Structuring Language (HyTime) extension to SGML.

1.3.3 Structure within and across web pages

Boyan et al. have observed that Web pages differ from general text in that they possess external and internal structure [5]. They use this information in Web tools to propagate rewards from interesting pages to those that point to them (also done by LaMacchia [21]) and to more heavily weight words in titles, headers, etc., when considering document/keyword similarity, a technique used earlier in Lycos by Mauldin and Leavitt [22]. Mauldin has made use of link information by naming a page with the anchor text of hyperlinks to it. Iwazume et al. have preferentially expanded hyperlinks containing keywords relevant to the user's query [16], although O'Leary observes that anchor text information can be unreliable [28]. LaMacchia has implemented or proposed heuristics similar to some mentioned in this paper, such as making use of the information in directory hierarchies [21]. Frei and Stieger have discussed how knowledge of the adjacency of nodes via hyperlinks can be used to help a user navigate or find the answer to a query [11].

1.4 Reader's Guide

Chapter 2 presents an ontology for the Web and Squeal, describing what types of information are made available to the Squeal user, more precisely defining the types of structural information available on the Web. Chapter 2 also provides examples of simple Squeal queries.

The Squeal language is defined in Chapter 3, including the algorithms used to implement the illusion that the Web is a relational database. These algorithms fetch and process information from the Web as implicitly required by user queries and would be useful in any domain where one wanted to provide a similar illusion by putting information in a database in such a lazy fashion.

The low-level details of the Squeal implementation, including how to add new relations, are presented in Chapter 4. While this chapter would be of value for modifying my code or reproducing the work, it contains no vital conceptual information and may be safely skipped by most readers.

Chapter 5 presents the three ParaSites that were implemented: a recommender system, a home page finder, and a moved page finder. For each application, the heuristics, Squeal code, and evaluation are provided. By demonstrating the simplicity and utility of structure-based applications expressed in Squeal, an argument is made for the value of the system.

Chapter 6 contains the conclusions drawn from this work, including a discussion of lessons learned and comparisons to existing work, as well as possible directions for future research.