Fall 1997

Structured document comparison and scientific data mining on the world wide web

Philip B. Johnson
New Jersey Institute of Technology

ABSTRACT

STRUCTURED DOCUMENT COMPARISON AND SCIENTIFIC DATA MINING ON THE WORLD WIDE WEB

by
Philip B. Johnson

The usefulness and accessibility of programs and systems have become important issues for users and researchers alike. A program's usefulness can be measured by the frequency with which it is used. From the author's or maintainer's point of view, the frequency of usage can be determined by how often a request for the software is received. In the past, a user became aware of a particular tool through various means, and contacted the author or maintainer to obtain a copy of it. This presented difficulties, ranging from language barriers to machine incompatibilities to control of the use of the program. Furthermore, accessibility of the program was limited to the particular machines on which it was installed (at the remote site). By porting programs and systems to the World Wide Web, the problems of accessibility and usefulness can be mitigated. Now programs can be advertised (in a non-commercial sense) to all interested parties, problems of machine incompatibility can be reduced (with the exception of browser incompatibilities), and control of the use and modification can be maintained. This thesis discusses the porting of two tools to the World Wide Web. The tools are SDISCOVER, a data mining tool used in protein string matching, and TREEDIFF, a structured document comparison toolkit. Spinoff of this research is the development of two home pages for conference registrations and Oracle user account applications in a university environment.

STRUCTURED DOCUMENT COMPARISON AND SCIENTIFIC DATA MINING ON THE WORLD WIDE WEB

by
Philip B. Johnson

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

Department of Computer and Information Science

October 1997

APPROVAL PAGE

STRUCTURED DOCUMENT COMPARISON AND SCIENTIFIC DATA MINING ON THE WORLD WIDE WEB

Philip B. Johnson

BIOGRAPHICAL SKETCH

Author: Philip B. Johnson

Degree: Master of Science in Computer Science

Date: October 1997

Undergraduate and Graduate Education:

- Master of Science in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 1997

- Bachelor of Science in Computer Science,
  Seton Hall University, South Orange, NJ, 1987

Major: Computer Science

ACKNOWLEDGEMENT

I would like to thank Dr. Jason T. L. Wang for his insightful discussions, guidance, and helpful pointers durings this project.

I would also like to thank George J. S. Chang for his suggestions on using Linux and Perl, and most of all for the use of his Web server on one of the NJIT DB Lab machines.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

This thesis discusses two tools which have been ported to the WWW for use there. The two tools are TREEDIFF, a structured document comparison toolkit, and SDISCOVER, a data mining tool for protein matching. The tools have been modified slightly in their I/O routines to support batch processing (i. e., non-interactive). Scripts in Perl and shell are used to handle the input of data and parameters for execution; typically, the source code for the tools has been modified to support output with HTML embedded (so the output looks presentable on the user's web browser). Note that no allowance has been made for those users with non-graphical browsers (e. g., Lynx). Neither the web pages nor the output for these tools have been aligned for these types of browsers.

1.1 Structured Document Comparison Using TREEDIFF

The structured document comparison toolkit, TREEDIFF, examines documents written using the SGML, or Standard Generalized Markup Language. A markup language is a specification on how to process files, whether they are documents, spreadsheets, or the like; individual markup instructions are known as 'tags'.

One type of markup language, known as descriptive markup, occurs when the document processing codes are embedded in the document itself. The codes for bolding or underlining in a document produced by a word-processing program are one such example. The codes used by utilities such as nroff and troff are another example. Some difficulties with these codes are that the processing commands might be platform specific, and that it is hard to compare two or more documents based solely on these embedded codes. Here is an example of markup in a document (this is nroff source for the Perl 5.003 man page):

```
.TH PERL 1 "perl 5.003 with" "25/Mar/96" "Perl Programmers Reference
Guide"
.IX Title "PERL 1"
.UC
.IX Name "perl - Practical Extraction and Report Language"
.if n .hy 0
.if n .na
.ds C+ C\v'-.1v'\h'-1p'\s-2+\h'-1p'+\s0\v'.1v'\h'-1p'
.de CQ          \" put $1 in typewriter font
.ft CW
'if n "\c
'if t \\&\\$1\c
'if n \\&\\$1\c
'if n \&"
\\&\\$2 \\$3 \\$4 \\$5 \\$6 \\$7
'.ft R
..
.\" @(#)ms.acc 1.5 88/02/08 SMI; from UCB 4.2
.  \" AM - accent mark definitions
.bd B 3
.  \" fudge factors for nroff and troff
.if n \{\
.  ds #H 0
.  ds #V .8m
.  ds #F .3m
.  ds #[ \f1
.  ds #] \fP
.\}
```

Another type of markup language is called generalized markup. This type is used to specify the structure of the document, rather than the specific actions to be carried out when processing the document [1]. Generalized markup allows for several improvements over descriptive markup:

- the preservation of information
- arbitrary processing instructions can be set for particular tags. This in turn provides flexibility and platform independence
- it is possible to make 'intelligent' queries on documents

SGML (Standard Generalized Markup Language) is a further abstraction of markup. It is a meta-syntax for developing other generalized markup languages. That is, it specifies the rules by which a variety of markup languages can be created [3]. The most well-known sub-type of SGML is HTML.

Here is an example of a document which has been marked-up using SGML (this is from a paper on extending DSSSL to handle trees) [2]:

```
<!DOCTYPE GCAPAPER PUBLIC "-//ATLIS//DTD GCAPAPER.DTD 19960213 Vers 1.0
//EN" "gcapaper1.dtd" >
<gcapaper>
<front>
<title>Semantic Extensions to DSSSL to Handle Trees
<author prime="1">
<fname>Matthew <surname>Fuchs
<address>
<affil>Walt Disney Imagineering
<aline>1401 Flower Street, P. O. B. 2502
<city>Glendale <state>CA <postcode>91221
<email>matt@wdi.disney.com
<Abstract>
<para>
We consider the syntax and semantics of the
<ACRONYM><TERM>TL</TERM><DD><PARA>Transformation
Language</PARA></DD></ACRONYM>
in the <ACRONYM><TERM>DSSSL</TERM><DD><PARA>Document Style Semantics
and  Specification Language</PARA></DD></ACRONYM>
specification<bibref>DSSSL96</bibref>.  At present
<ACRONYM><TERM>TEs</TERM><DD><PARA>Transformation
Expressions</PARA></DD></ACRONYM>
are less than first-class language objects - they
must all reside at the top level, and cannot be manipulated like other
DSSSL/Scheme objects.  In particular, there is no means of passing
information among TEs, so one TE cannot take advantage of information
derived by another, such as passing data about parent nodes to direct
```

```
the transformation of child nodes.  We propose extending the DSSSL
syntax to allow a DSSSL program to better exploit the tree-like nature
of the source grove by providing a semantics for nesting query
expressions, allowing information to be passed around while retaining
DSSSL's functional nature.  The TEs would also come closer to being
first-class objects.  We suggest these extensions will make DSSSL
programs easier to write and probably easier to optimize.
</para>
```

The expression, "document comparison," is to mean the detection of change of the *structure* of the document. This differs from a tool such as the Unix *diff* utility, which considers the file to be a string of characters. *Diff* will examine pairs of files, comparing strictly character-by-character. If the markup within either file is different from the other, then *diff* will report that as a difference (and the same goes for regular text within the files, as well). TREEDIFF compares the structure of files by first translating each file into an ordered labeled tree. The structure of the tree is determined by the structure of the markup in the file. After this translation process, the trees are compared using approximate tree matching techniques. These techniques will find a minimum quantity of edit operations (insert, relabel, and delete node) which are required to transform one tree to the other. Furthermore, TREEDIFF can find portions of one document that are common to the other.

Document comparison is useful in several industries; among them are defense, aerospace, and publishing. It is also useful for comparing documents marked up with HTML and LaTeX. For example, it is possible to learn if two HTML pages are different, or for detecting plagiarism within two LaTeXdocuments. Detecting structural changes in documents is important in the areas of data warehousing, digital libraries, hypermedia, and Internet databases [1].

1.2 Scientific Data Mining Using SDISCOVER

In data mining, there can be several applications. One typically is analyzing a database for similarities among and between records, as in purchasing habits or the likelihood of a prospective credit card applicant to default on their payment based on their purchasing and payment histories, as well as other factors. Another type of data mining, the type discussed here, is the discovery of structural patterns in scientific data [4]. That is to say, a single record of scientific data is examined with the intent to discern some pattern in that record; then other records are searched for that same pattern.

Specifically, the scientific data which is investigated for pattern discovery are protein strings. The protein strings are strings of amino acids; 20 amino acids are represented by a single capital letter. Here is an example of a pair of sequences [4]:

```
>RASL_MOUSE TRANSFORMING PROTEIN P21/K-RAS 2B.
MTEYKLVVVGAGGVGKSALTIQLIQNHFVDEYDPTIEDSYRKQVVIDGETCLLDILDTAG
QEEYSAMRDQYMRTGEGFLCVFAINNTKSFEDIHHYREQIKRVKDSEDVPMVLVGNKCDL
PSRTVDTKQAQELARSYGIPFIETSAKTRQGVDDAFYTLVREIRKHKEKMSKDGKKKKKK
SRTRCTVM
>RASN_HUMAN TRANSFORMING PROTEIN P21/N-RAS.
MTEYKLVVVGAGGVGKSALTIQLIQNHFVDEYDPTIEDSYRKQVVIDGETCLLDILDTAG
QEEYSAMRDQYMRTGEGFLCVFAINNSKSFADINLYREQIKRVKDSDDVPMVLVGNKCDL
PTRTVDTKQAHELAKSYGIPFIETSAKTRQGVEDAFYTLVREIRQYRMKKLNSSDDGTQG
CMGLPCVVM
```

The benefit of this work is that if two seemingly dissimilar strings of proteins have in them a particular pattern, then perhaps these two strings perform the same function [4]. The patterns discovered are regular expressions of the form $*X_1*X_2*...$, where X_1, X_2, X_n, are segments of a sequence, and '$*$' represents a Variable Length Don't Care (VLDC). In matching the regular expression $*X_1*X_2*...$ (to be read as: VLDC 'sequence X_1' VLDC 'sequence X_2' VLDC...) with some other sequence S, the VLDCs may substitute for zero or more letters in S at zero cost.

For example, consider the three sequences shown below [4]:

S_1 : `YDPMIEDKEYSRLVG`

S_2 : `RMKQLGRTYDPAVWG`

S_3 : `YDPMNWNFEKETLVG`

Suppose only exactly coinciding segments of lengths greater than 3 are considered as 'similar'. Then S_1 and S_3 have one similarity (or common pattern):

$$*S_1[1,4]* = *\mathrm{YDPM}* \Longleftrightarrow *\mathrm{S}_3[1,4]* = *\mathrm{YDPM}*$$

where for some sequence V, V_n[x,y] is a segment of sequence V_n from the x$^{\text{th}}$ letter to the y$^{\text{th}}$ letter inclusively. If similarities within distance one are sought, i. e., one mutation, mismatch, insertion, or deletion, then S_1, S_2, and S_3 share three similar patterns:

$$\begin{aligned} & *S_1[1,4]* = *\mathrm{YDPM}\ * \\ \Longleftrightarrow\ & *S_2[8,11]* = *\mathrm{TYDP}\ * \\ \Longleftrightarrow\ & *S_2[9,12]* = *\mathrm{YDPA}\ * \\ \Longleftrightarrow\ & *S_3[1,4]* = *\mathrm{YDPM}* \end{aligned}$$

If similarities having the form $*X*Y*$ are sought with lengths greater than τ and one mutation allowed, then S_1 and S_2 share the following four similar patterns:

$$\begin{aligned} & *S_1[1,4]*S_1[12,15]* = *\mathrm{YDPM} * \mathrm{RLVG}\ * \\ \Longleftrightarrow\ & *S_1[1,5]*S_1[13,15]* = *\mathrm{TYDP} * \mathrm{LVG}\ * \\ \Longleftrightarrow\ & *S_3[1,4]*S_3[12,15]* = *\mathrm{YDPA} * \mathrm{TLVG}\ * \\ \Longleftrightarrow\ & *S_3[1,5]*S_3[13,15]* = *\mathrm{YDPM} * \mathrm{LVG}\ * \end{aligned}$$

End of example

CHAPTER 2

STRUCTURED DOCUMENT COMPARISON ON WWW

The LaTeX/SGML document comparison pages have the following fields:

```
FIELD NAME                                             TYPE
==========                                             ====
URL 1                                                  text, scrollable
URL 2                                                  text, scrollable

<submit> <clear> buttons
```

Field list ends

The steps that occur in using the web versions of the document comparison tools are as follows:

1. User fills out preliminary form, at one of the two following addresses:

   for LaTeX use `http://www.cis.njit.edu/~discdb/clatex.html`

   See figure B.1, page 45.

   for SGML, use `http://point.njit.edu:8000/~discdb/csgml.html`

   See figure C.1, page 62.

2. Each form has two fields. Each field is to have a fully-qualified URL typed in. This will retrieve the document that the URL points to.

3. Each file that is to be retrieved must be visible by the browser. That is, if web-browsable documents or files must be placed under the, say, *public_html* directory, then the documents to be retrieved must also be similarly placed.

4. User clicks on '`submit`'. This calls '`getlatex.cgi`', a Perl script. `getlatex.cgi` reads the data input by the user, retrieves the files specified, and writes them

to temporary files. The retrieval process is performed by the '`getstore`' function which is part of the LWP set of Perl modules for web programming.

If an error occurs while retrieval of the documents is being attempted, an appropriate error message (from HTML) is displayed. If the documents cannot be found, the an explanatory message is displayed, indicating that the files must be visible by the browser. For LaTeX see figure B.4, page 48. For SGML, see figure C.4, page 65.

Otherwise, a notice of successful retrieval is displayed, and the user is instructed to choose some document comparison options. For LaTeX see figure B.2, page 46. For SGML, see figure C.2, page 63.

5. When the user continues, the script '`dolatex.sh.cgi`' (a shell script) is called. This script extracts the two document comparison options and then calls a command line version of the TREEDIFF program.

   The output is displayed on the user's browser in preformatted style; the output is not saved to any temporary file. For LaTeX see figure B.3, page 47. For SGML, see figure C.3, page 64.

CHAPTER 3

SCIENTIFIC DATA MINING ON WWW

The Protein sequence discovery tool (SDISCOVER) page has the following fields:

```
FIELD NAME                                              TYPE
==========                                              ====
user's name                                             text, scrollable
user's e-mail address                                   text, scrollable
user's web site (optional)                              text, scrollable
list of sequences                                       text box, scrollable
motif type                                              radio buttons
allowed minimum length of motif                         text, scrollable
allowed minimum occurrence number                       text, scrollable
quantity of allowed matches                             drop-down list

<submit> <clear> buttons
```

Field list ends

The steps that occur in using the web version of the protein discovery tool are as follows:

1. User fills out preliminary form at

   `http://www.cis.njit.edu/~discdb/homer.pro4.html`

   (See figures A.1 and A.2, pages 30 and 31, respectively), and clicks on `'submit'`.

2. This calls `homer.pro4.cgi`, a Perl script. `homer.pro4.cgi` reads the data input by the user and returns to the user the data that will be submitted to the discovery program. The form asks the user for confirmation to continue processing (see fig A.3, page 32). `homer.pro4.cgi` also writes two temporary files for use by the discovery program. The first file consists of the set of sequences, while the second file is a list of analysis parameters (the motif

type, the minimum length, the occurrence number, and the quantity of allowed matches).

3. If the user agrees to continue, the shell script `homer.pro2.sh.cgi` is called. This script executes a command-line version of the discovery tool. The command-line version reads the two previously-mentioned files of data and parameters as input, and writes the output to a temporary file. This temporary file is in turn used as input to the ssort routine. The sorting routine displays the result to the user's browser with embedded html formatting (see fig A.4, page 33).

The Perl cgi script, `homer.pro4.cgi`, does the following:

1. The method used by the form is `'post'`; therefore, the input data is read from stdin.

2. The Perl script reads the data supplied via the post mechanism into a variable, which is then split into substrings delimited by the `'&'` character. These substrings are further broken down into keys and their associated values delimited on the `'='` character. Any escaped character in the value part of the string is converted to its hexadecimal representation. The values are assigned to a hash using the key as the index. If the key already has a value assigned to it, the new value is appended to the old with `'\0'` as the delimiter. If the key does not have a value assigned, then a new entry is made into the hash.

3. Some error checking is performed prior to executing the discovery tool. The errors to look for are: no sequences supplied; motif type not 1 or 2[*] (should be impossible because this is selected via radio buttons); minimum length of sequences is non numeric or zero; minimum occurrence is non-numeric or zero.

[*]Note that `'motif type'` does not have a default value.

Also, the sequences are cleaned up: any lower-case letters are translated to upper case, and any non-alphabetic characters (excluding newline) are stripped out.

Originally, the script was going to perform other checking of the supplied sequences, such as incorrect placement of comments, but the decision was made to assume that the user is knowledgeable about such things. Therefore, any further error error checking is performed in the discovery tool itself during execution.

The script also writes the data (the cleaned sequences and the other parameters) to temporary files.

After writing the temporary files, the script execution proceeds as above in step 2.

CHAPTER 4

RELATED WORK

Items related to the porting of SDISCOVER and TREEDIFF to the World Wide Web include the development of an on-line registration page for the Very Large Database (VLDB) conference, to be held in August of 1998 in New York City, and the development of an online account registration (in a university environment) for the Oracle database system.

4.1 Very Large Database Registration

The VLDB Registration page registers a participant for the VLDB Conference, to be held in August of 1998 in New York City.

The registration page has the following fields:

FIELD NAME	TYPE
Last Name	text box, scrollable
First Name	text box, scrollable
First Name	text box, scrollable
title	radio buttons
Dr	
Mr	
Mrs	
Ms	
Prof	
Company/Institution	text box, scrollable
Address 1	text box, scrollable
Address 2	text box, scrollable
Address 3	text box, scrollable
City	text box, scrollable
State/Province	text box, scrollable
Country	text box, scrollable
Telephone Number	text box, scrollable
Facsimile Number	text box, scrollable

Field list continued on next page

Internet E-Mail Address	text box, scrollable
Payment Method	radio buttons
Check	
Int'l Money Order	
Charge to VISA	
Charge to MasterCard	
Charge to Amex	
Charge to Diner's Club	
Advance Registration	radio buttons
Member	
NonMember	
Student	
Tutorials for Advance Registration	radio buttons
One Member/NonMember	
Two Member/NonMember	
Three Member/NonMember	
Four Member/NonMember	
Late/On site Registration	radio buttons
Member	
NonMember	
Student	
Tutorials for Late/On site Registration	radio buttons
One Member/NonMember	
Two Member/NonMember	
Three Member/NonMember	
Four Member/NonMember	
Tutorial Selection	radio buttons
Tutorial Number 1	
Tutorial Number 2	
Tutorial Number 3	
Tutorial Number 4	
Tutorial Number 5	
Tutorial Number 6	
Quantity of banquet tickets	text box, scrollable
Total for banquet	text box, scrollable
IEEE Member number	text box, scrollable
Check/Money Order Amt	text box, scrollable
Credit Card Amt	test box. scrollable
Name on Credit Card	text box, scrollable
Type of Credit Card	radio buttons
VISA	
MasterCard	
AMEX	
Diner's Club	
Credit Card Number	text box, scrollable

Field list continued on next page

```
Expiration Date                      text box, scrollable

<submit> and <reset> buttons
```

Field list ends

The cgi program does the following:

1. The method used is `'post'`; therefore, the input data is read from stdin.

2. Read the input and place it in the shell variable `'line'`. This will be a temporary working/storage area that will allow extraction of the items of interest. Strip the trailing carriage return.

3. Data is extracted from the working variable `'line'` in much the same manner that data is removed from a stack.

   The first field on the line, delimited by the character `'&'`, is extracted (popped) and stored in a shell variable `'work'`. The remainder of the line gets moved up one field position, so that the second field is now at the first position. The popped data item is further manipulated to extract the value, delimited in this case by the character `'='`. The code used is as follows:

   ```
   # pop off first field...
   work=`echo "${newline}" | cut -d'&' -f1`

   # move line up one field pos....
   newline=`echo "${newline}" | cut -d'&' -f2-`

   # extract name of field...
   name=`echo "${work}" | cut -d= -f1`

   # extract value of field...
   value=`echo "${work}" | cut -d= -f2-`
   ```

4. The registration information is extracted from the input line linearly; that is to say, the last name is the first item on the string, so it's extracted first. The

value of the last name is stored in a program variable of an appropriate name, usually the item itself (e. g., for last name, the name of the shell variable is `'lname'`). The first name is the next item on the string, so it's extracted next. And so on....

In each case, the value of the extracted string is compared to the null string. If the value is null, an appropriate error message is displayed indicating that none of the fields can be left blank. Furthermore, the offending field is highlighted.

5. Due to the sequential nature of the way in which the data is extracted from the working variable `'line'`, blanks (i. e., fields on the form left empty) are not permitted. If a field is left unfilled, then all the data which follows this particular field will not be assigned to the correct shell variable. To guard against this, if a field is left blank, the script will detect this and issue an appropriate error message. Note that entering spaces into a field is the same as entering valid data.

6. The information concerning the tutorials requires special mention. The difficulty is that a registrant is allowed to register and attend a maximum of four tutorials, but the form will allow the registrant to sign up for all six tutorials. The script deals with this by assigning each tutorial a number, specifically the power of two. Thus, tutorial number 1 has the value 1. Tutorial number 2 has the value 2, tutorial number 3 has 4, tutorial number 4 has 8, number 5 has 16, and number 6 has 32. All six tutorial choices are extracted from the input line, unchosen tutorials are assigned the value 0, and their values are added together. The sum of all the tutorials indicates which tutorials are desired. If the sum of all the desired tutorials is equal to certain values, then it is known that the registrant has requested more than the allowed maximum. If the registrant has chosen five tutorials, the values of interest are: 31, 47, 55,

59, 61, 62. If the registrant has chosen all six tutorials, the value of interest is 63. If the registrant has chosen 5 or 6 tutorials, an appropriate error message is displayed.

7. The next two steps deal with files created or appended to to facilitate record keeping. The files are written to specific directories under control of the programmer. This is due to the security restrictions on writing files to other people's directories. If another programmer assumes maintenance of the script, then the specific directories will have to be changed to reflect the change in personnel.

8. An internal-use only file is created or appended to. This file is used to check if the user has already registered. Pipe symbols are used to delimit the fields. It is assumed that each individual has a unique email address. Thus, checking for a previous registration is done via the e-mail address field.

9. Next, the external registration file is created or appended to. This file is sorted on last name of the registrant and e-mailed to the registration administrator. Also created is a simple list of registrants, beginning with the first registrant and ending with the last. This file has appropriate labels for each field. This file over writes the previous list of registrants each time it is created. The reason for creating the file anew from oldest to newest is to present the registration administrator with the most current list of all registrants. This file is e-mailed to the registration administrator.

4.2 Oracle Registration

The Oracle registration page registers two students for accounts on the Oracle database system. Each pair of students is considered a group. The registration page has the following fields:

```
FIELD NAME                              TYPE
=================                       ===================
Student Id                              text box, not scrollable,
                                           no punctuation allowed
First Name                              text box, scrollable
Last Name                               text box, scrollable
Course Number                           text box, not scrollable
Section Number                          text box, not scrollable
E-Mail Address                          text box, scrollable,
                                           punctuation allowed
Telephone Number                        text box, scrollable,
                                           punctuation allowed
Physical Mailing Address                text box, scrollable,
                                           information to be entered
                                           all on one line

<submit> and <reset> buttons
```

Field list ends

The entire set of fields is duplicated for the other student.

The cgi program does the following:

1. The method used is `'post'`; therefore, the input data is read from stdin.

2. Read the input and places it in the shell variable `'line'`; strip the trailing carriage return.

3. The registration info is extracted from the input line linearly; that is to say, the student id is the first item on the string, so it's extracted first. The value of the student id is stored in a shell variable of an appropriate name, usually the item itself (e. g., for student id, the name of the shell variable is `'student_id'`).

The first name is the next item on the string, so it's extracted next. And so on....

In each case, the value of the extracted string is tested to see if the value is the null string. If the value is null, an appropriate error message is displayed indicating that none of the fields can be left blank. Furthermore, the offending field is highlighted. In this form, blank fields are not permitted.

4. The next two steps deal with files created or appended to to facilitate record keeping. The files are written to specific directories under control of the programmer. This is due to the security restrictions on writing files to other people's directories. If another programmer assumes maintenance of the script, then the specific directories will have to be changed to reflect the change in personnel.

5. An internal-use only file is created or appended to. This file is used to check if the user has already registered. Pipe symbols are used to delimit the fields. Checking is done via the student id field.

6. Next, the external registration file is created or appended to. This file is sorted on last name of the registrant and e-mailed to the course instructor, so that it can be passed on to the Oracle DBA. Also created to is a simple list of registrants, beginning with the first registrant and ending with the last. This file has appropriate labels for each field. This file over writes the previous list of registrants. The reason for writing the file anew from oldest to newest is to present the course instructor with the most current list of all registrants. This file is also e-mailed to the registration administrator.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

The tools, SDISCOVER and TREEDIFF, were ported to the World Wide Web to make them more accessible to interested users. In order to maintain the functionality of the standalone versions in the web-based versions, various types of scripts (Perl and shell) were used. In addition, the programs themselves were modified to accept input data on the command line and to output data with embedded HTML codes.

One difficulty that arose was the restriction placed on non-root users of the web server. This is due to the security measures put in place (which are typically used in a networked environment). In the case of the SGML file comparison system, only the root user can open a file in the */tmp* directory (for a Unix system). This problem was surmounted by running a user server (that is to say, a non-root server). This is why the URL for this comparison tool specifies port 8000. However, when running a non-root server a problem can arise if the machine on which the server is running is re-booted; then the server must be restarted. Until this non-root server is restarted, the particular system (i. e., SGML file comparison) will not execute. A possible work-around of this restriction would be to run a standalone web server. However, this requires the acquisition of an Internet connection, an Internet address, and other various items.

Further work includes the development of web site change detection, which will involve unordered tree matching, and the inclusion of different DTDs and catalog files for SGML comparison.

APPENDIX A

RESOURCES FOR SDISCOVER ON WWW

A.1 INTRODUCTION

This appendix contains the following resources used in implementing SDISCOVER on the World Wide Web:

1. Perl and shell scripts

2. Screen shots of SDISCOVER

3. A tutorial on using SDISCOVER on the WWW

A.2 SCRIPTS

This section contains the scripts[†] used to implement SDISCOVER on the World Wide Web. There are two scripts used:

1. `homer.pro4.cgi` - a Perl 5.003 script

2. `homer.pro2.sh.cgi` - a Bourne shell script

A.2.1 homer.pro4.cgi

Following is the Perl 5.003 script used to process the user data, prior to execution of SDISCOVER.

```
#!/home/discdb/local/bin/perl
#  library on homer
#  !/opt/local/bin/perl
#  library on homer
#  !/usr/bin/perl5.003
```

[†]N. B.: The text of this and the following scripts has been modified slightly. The modifications are primarily in the form of newlines inserted where they otherwise would not be. These modifications are imposed by the requirements of the thesis format.

```
#  library on darkstar

$webmaster = "phil\@homer";
$gateway = "CGI Discover Gateway [v1.0]";
$MAXSEQ = 200;
$MAXLENGTH = 5000;
$space = " ";
#$tempseqfile = "/tmp/disco.seq";
$tempseqfile = "/home/discdb/public_html/disco.seq";
#$tempparmfile = "/tmp/disco.parms";
$tempparmfile = "/home/discdb/public_html/disco.parms";

&parse_data (*PROTEIN);

# what do we have here?
#
# variable names:
#
# $PROTEIN{'name'} - - - - sender's name
# $PROTEIN{'email'} - - -  sender's email adress
# $PROTEIN{'url'} - - - -  sender's URL (optional)
# $PROTEIN{'sequence'} - - list of sequences
# $PROTEIN{'min_length'} - minimum length of sequences
# $PROTEIN{'min_occur'} -  minimum # of occurrances
# $PROTEIN{'allowed'} - -  number of allowed mutations values 1 - 10
# $PROETIN{'type'} - - - - search type: *X* or *X*Y* values 1, 2
#
############################
#
# the following stuff displays the parsed output from the browser
# this should be replaced with error-checking code & then a call
# to the discovery program
#
# error conditions (derived from the discovery source)
#
# 1. no sequences supplied
# 2. motif type not 1 or 2
# 3. minimum length of interesting motifs not numeric
# 4. minimum occurance number not numeric
# 5. number of mutations allowed


# count number of sequences...
```

```
$num_seqs = $PROTEIN{'sequence'} =~ s/\>/\>/gs; # tr/\>/\>/;

if ($num_seqs eq 0) {
   &return_error (666, "No sequences",
      "No protein sequences were supplied!");
}


# see if motif type is 1 or 2...

if ($PROTEIN{'mtype'} eq 1) {
   $type = "*X*";
}
elsif ($PROTEIN{'mtype'} eq 2) {
   $type = "*X*Y*";
}
else {
   &return_error (888, "bad motif",
      "Chosen motif type must be 1 or 2!");
}


# minimum length is numeric & not zero?;

$PROTEIN{'min_length'} =~ s/\D//;
$min_len = $PROTEIN{'min_length'} =~ tr/0-9//;

if ($min_len eq 0) {
   &return_error (444, "Error with Interesting Motifs",
      "Minimum length of interesting motifs is not numeric!");
}
elsif ($PROTEIN{'min_length'} eq 0) {
   &return_error (444, "Length = 0",
      "Minimum lenth of interesting motifs is 0!");
}


# minimum occurance is numeric & not zero? ;

$PROTEIN{'min_occur'} =~ s/\D//;
$min_occ = $PROTEIN{'min_occur'} =~ tr/0-9//;

if ($min_occ eq 0) {
   &return_error (333, "Error with Minimum Occrrance",
      "Minimum occurrance not numeric!");
}
elsif ($PROTEIN{'min_occur'} eq 0) {
   &return_error (333, "Error with Minimum Occurrence",
      "Minimum occurrence is 0!");
}


# check for allowed mutations to be between 1 & 10 ...;
# should never happen: 'select' html construct should do this
# automatically;

$PROTEIN{'allowed'} =~ s/\D//;

#if (($PROTEIN{'allowed'} lt 1) or ($PROTEIN{'allowed'} gt 10)) {
#   &return_error (222, "Bad match allowance",
#      "Quantity of allowed matches not between 1 and 10
#       inclusive!");
#}


# clean up sequences...
#
# 1. strip out non-letters
# 2. translate to upper case;

$PROTEIN{'sequence'} =~ tr/a-zA-Z\n\>//cd;
$PROTEIN{'sequence'} =~ tr/a-z/A-Z/;


unless (open (FILE1, ">" . $tempseqfile)) {
   &return_error (234, "bad open", "Cannot open `", $tempseqfile, "'
                  for writing");
}
flock (FILE1, 2);
print FILE1 $PROTEIN{'sequence'};
flock (FILE1, 8);
close (FILE1);

unless (open (FILE2, ">" . $tempparmfile)) {
   &return_error (234, "bad open", "Cannot open `", $tempparmfile, "'
                  for writing");
}
```

```
flock (FILE2, 2);
print FILE2 $PROTEIN{'mtype'};
print FILE2 $space;
print FILE2 $PROTEIN{'min_length'};
print FILE2 $space;
print FILE2 $PROTEIN{'min_occur'};
print FILE2 $space;
print FILE2 $PROTEIN{'allowed'};
flock (FILE2, 8);
close (FILE2);


unless (open (FILE, $tempseqfile)) {
  &return_error (123, "bad file", "Cannot open `", $tempseqfile, "'
                  for reading");
}
flock (FILE, 2);
$stuff = <FILE>;
flock (FILE, 8);
close (FILE);

print "Content-type: text/html\n";

print <<End_Of_Form;

<html>

<head>
<title>Protein Sequence Discovery Tool</title>
</head>

<h3>Here's what you're going to send:</h3>

<hr>

<table>
<tr><td align=right>Your Name:              <td><b>$PROTEIN{'name'}
                  </b></td></tr>
<tr><td align=right>Your e-mail address:    <td><b>$PROTEIN{'email'}
                  </b></td></tr>
<tr><td align=right>Your web site (optional)<td><b>$PROTEIN{'url'}
                  </b></td></tr>
</table>
<p>
```

```
<hr>

<h4>Your sequences are shown below.</h4>

<pre>
$PROTEIN{'sequence'}
</pre>

<hr>

<p>
The form of the motifs:&nbsp;<b>$type</b>
<p>
The minimum length of interesting motifs:&nbsp;<b>$PROTEIN
                  {'min_length'}</b>
<p>
The minimum occurrence number:&nbsp;<b>$PROTEIN{'min_occur'}</b>
<p>
The quantity of allowed mutations:&nbsp;<b>$PROTEIN{'allowed'}</b>

<hr>

<h4>To execute the Discovery Program, click on <b>Execute</b></h4>
<!-- prof wang wants to show only sorted output! 7-29-97 -->
<!-- in that case, use 'homer.pro2.sh.cgi' in place of   -->
<!--                   'homer.pro1.sh.cgi'               -->
<form method=get action="homer.pro2.sh.cgi">
<input type="submit" value="Execute">
<!-- <input type="reset" value="Return to Input Form"> -->
<h4>To return to input form. click on Browser Return Button</h4>
</form>

</body>
</html>


End_Of_Form

###########################

sub parse_data
{
   local (*FORM_DATA) = @_;  # any args passed to an array come
```

```
                              # in via @_

   # local variables:
   local ( $request_method,  # scalar
           $query_string,    # scalar
           @key_value_pairs, # associative array (aka hash)
           $key_value,       # scalar
           $key,             # scalar
           $value            # scalar
         );

   $request_method = $ENV{'REQUEST_METHOD'};  # get delivery method

   if ($request_method eq "GET") {
      $query_string = $ENV{'QUERY_STRING'};  # if 'get', read
                                             # environment var
   }
   elsif ($request_method eq "POST") {
      read (STDIN, $query_string, $ENV{'CONTENT_LENGTH'});
                                             # if 'post', read
                                             # from stdin
   }
   else {
      &return_error (500, "server error", "Server uses unsupported
                    method.");
   }

   @key_value_pairs = split (/&/, $query_string);  # split string on
                                                   # '&' boundaries

   foreach $key_value (@key_value_pairs) {
      ($key, $value) = split (/=/, $key_value);
      $value =~ tr/+/ /;
      $value =~ s/%([\dA-Fa-f0-9][\dA-Fa-f0-9])/pack ("C", hex
                 ($1))/eg;

      if (defined($FORM_DATA{$key})) {
         $FORM_DATA{$key} = join ("\0", $FORM_DATA{$key}, $value);
      }
      else {
         $FORM_DATA{$key} = $value;
      }
   }
}
```

```
###########################

sub return_error
{
   local ($status, $keyword, $message) = @_;

   print "Content-type: text/html", "\n\n";
   print "Status: ", $status, " ", $keyword, "\n\n";

   print <<End_of_Error;

<head>
   <title>CGI Program - Unexpected Error! Status: $status </title>
</head>
<body>
   <h1>$keyword</h1>
   <h2>$message</h2>
   Please contact $webmaster for more information.
</body>

End_of_Error

   exit (1);

}

###########################

sub return_thanks
{
  if ($PROTEIN{'url'}) {
      print "location: ", $PROTEIN{'url'}, "\n\n";
   }

   else {
      print "Content-type: text/html\n\n";

      print <<Thanks;

<html>
<title>$gateway</title>
<body>
<h1>Thanks!</h1>
</body>
</html>

Thanks
   }
}
```

A.2.2 homer.pro2.sh.cgi

Following is the Bourne shell script used to execute the command line version of the discovery program.

```
#!/bin/sh

#touch /home/discdb/public_html/protein.final.html
#chmod 755 /home/discdb/public_html/protein.final.html
echo "Content-type: text/html\n\n"

/home/discdb/public_html/cgi-bin/discover4.cgi \
   /home/discdb/public_html/disco.parms
   /home/discdb/public_html/disco.seq

/home/discdb/public_html/cgi-bin/homer.ssort.cgi \
   /home/discdb/public_html/cgi-bin/data.out

#cp final.html ../protein.final.html
#chmod 644 /home/discdb/public_html/cgi-bin/final.html
#echo "Location: http://www.cis.njit.edu/~discdb
       /cgi-bin/final.html\n"
```

A.3 SCREEN SHOTS FOR SDISCOVER

Following are the screen shots of SDISCOVER on the WWW. The first two are of the initial page (it's too large to show all on one screen shot), the third is the confirmation screen, and the fourth is the results screen. The last is an example of how sequence strings should be entered.

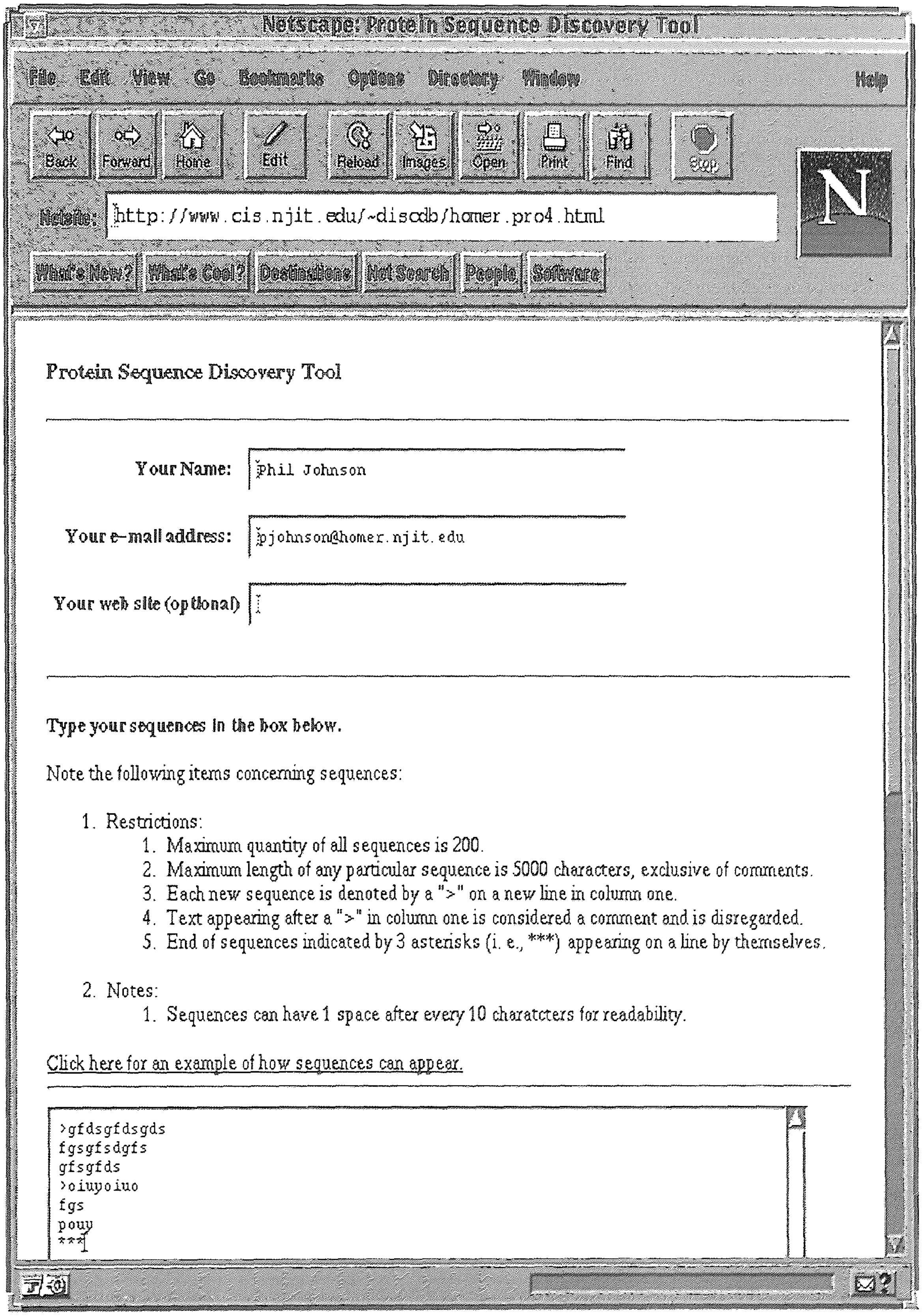
1. First Half of SDISCOVER Page, Fig. A.1

2. Second Half of SDISCOVER Page, Fig. A.2

3. Confirmation for SDISCOVER, Fig. A.3

4. Output From SDISCOVER, Fig. A.4

5. Example of Protein Sequence Strings, Fig. A.5


Netscape: Protein Sequence Discovery Tool
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find Stop
What's New? What's Cool? Destinations Net Search People Software

Netsite: http://www.cis.njit.edu/~discdb/homer.pro4.html

Protein Sequence Discovery Tool

Your Name:	Phil Johnson
Your e-mail address:	pjohnson@homer.njit.edu
Your web site (optional)	

Type your sequences in the box below.

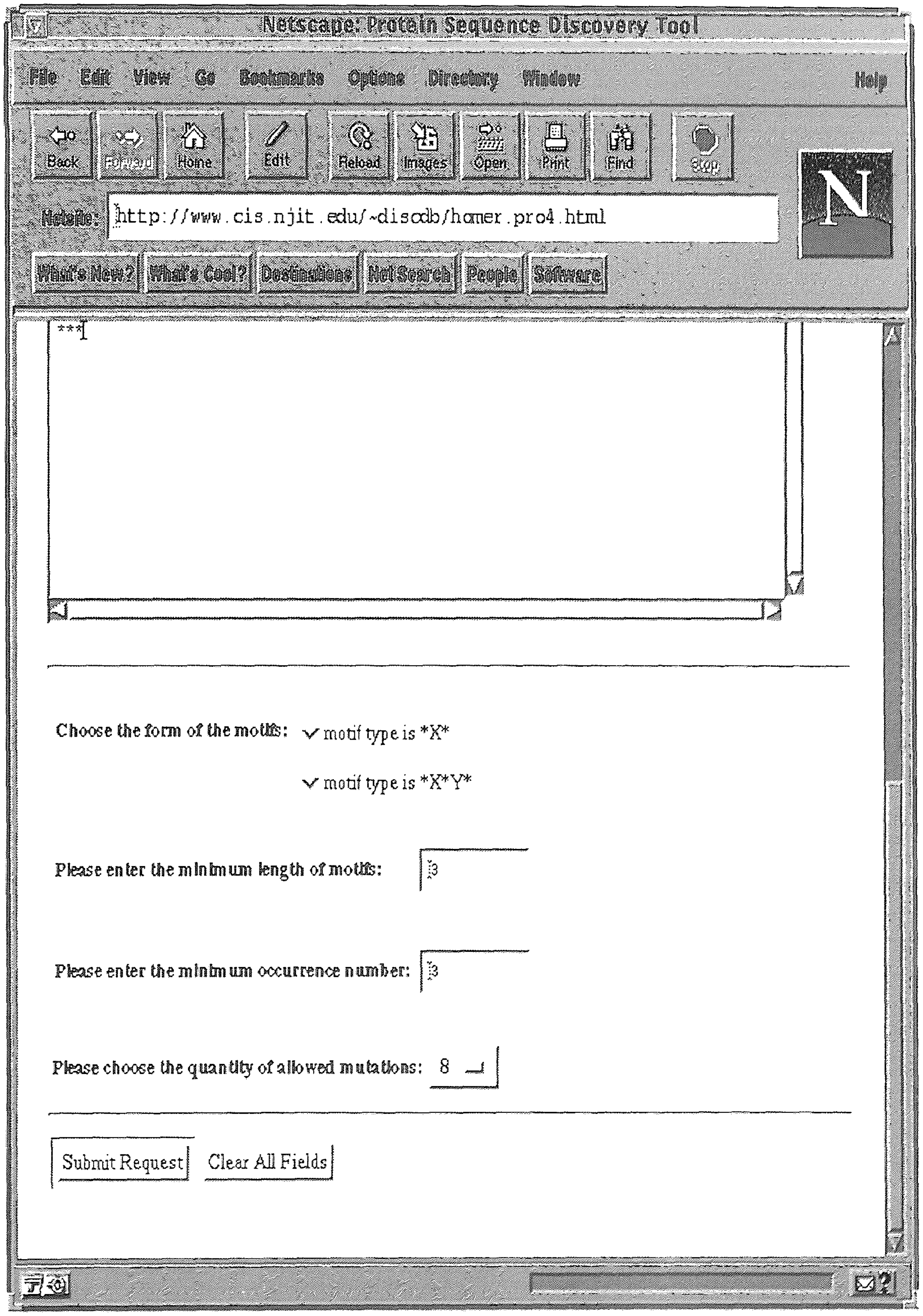
Note the following items concerning sequences:

1. Restrictions:
   1. Maximum quantity of all sequences is 200.
   2. Maximum length of any particular sequence is 5000 characters, exclusive of comments.
   3. Each new sequence is denoted by a ">" on a new line in column one.
   4. Text appearing after a ">" in column one is considered a comment and is disregarded.
   5. End of sequences indicated by 3 asterisks (i. e., ***) appearing on a line by themselves.
2. Notes:
   1. Sequences can have 1 space after every 10 charatcters for readability.

Click here for an example of how sequences can appear.

```
>gfdsgfdsgds
fgsgfsdgfs
gfsgfds
>oiupoiuo
fgs
pouy
***
```

Figure A.1 First Half of SDISCOVER Page



http://www.cis.njit.edu/~discdb/homer.pro4.html

***

Choose the form of the motifs: ✓ motif type is *X*

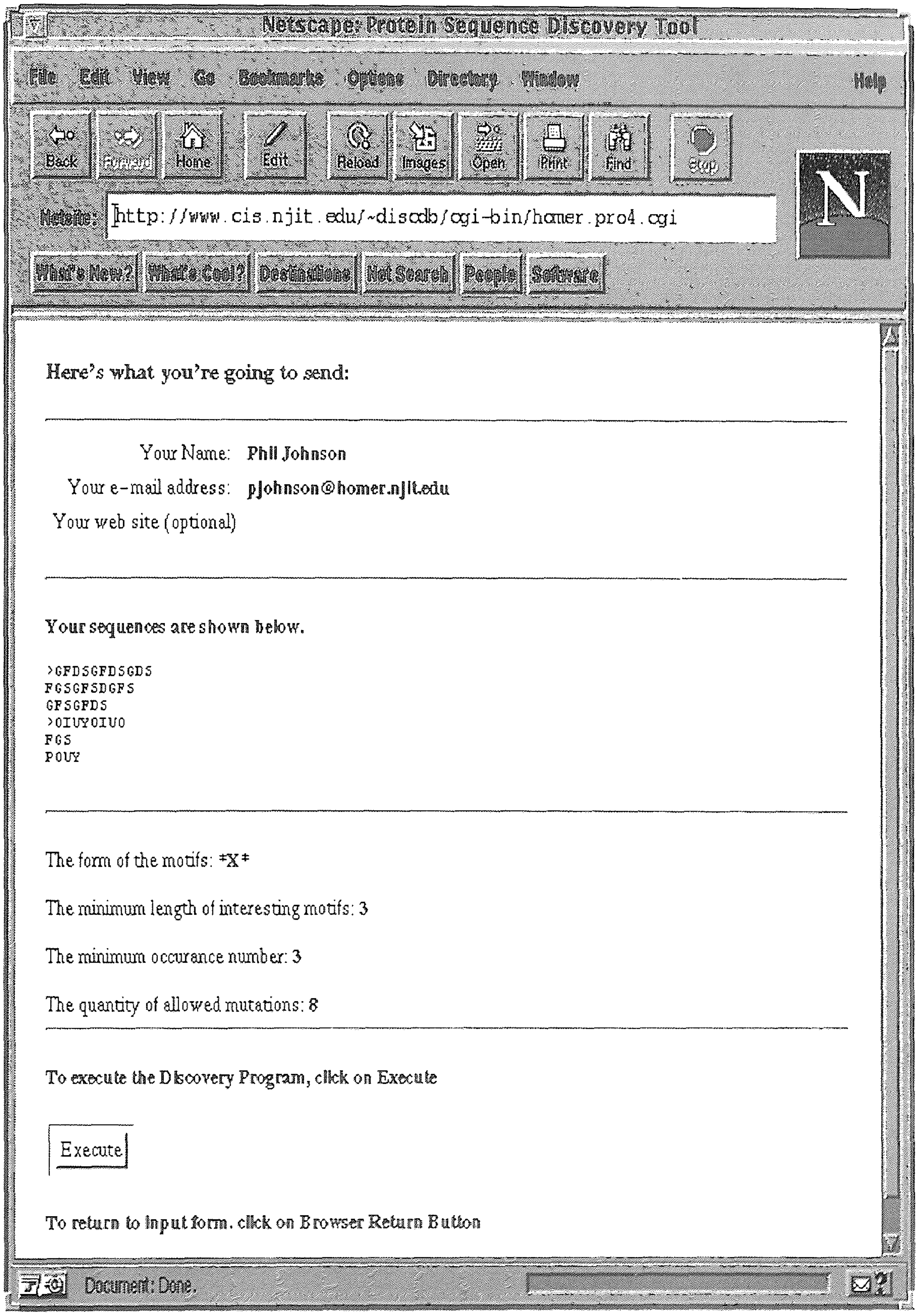
✓ motif type is *X*Y*

Please enter the minimum length of motifs: 3

Please enter the minimum occurrence number: 3

Please choose the quantity of allowed mutations: 8

Figure A.2 Second Half of SDISCOVER Page


Netscape: Protein Sequence Discovery Tool
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find Stop
Netsite:
What's New? What's Cool? Destinations Net Search People Software
Document: Done.

http://www.cis.njit.edu/~discdb/cgi-bin/homer.pro4.cgi

Here's what you're going to send:

Your Name: **Phil Johnson**

Your e-mail address: **pjohnson@homer.njit.edu**

Your web site (optional)

Your sequences are shown below.

```
>GFDSGFDSGDS
FGSGFSDGFS
GFSGFDS
>OIUYOIUO
FGS
POUY
```

The form of the motifs: *X*

The minimum length of interesting motifs: 3

The minimum occurance number: 3

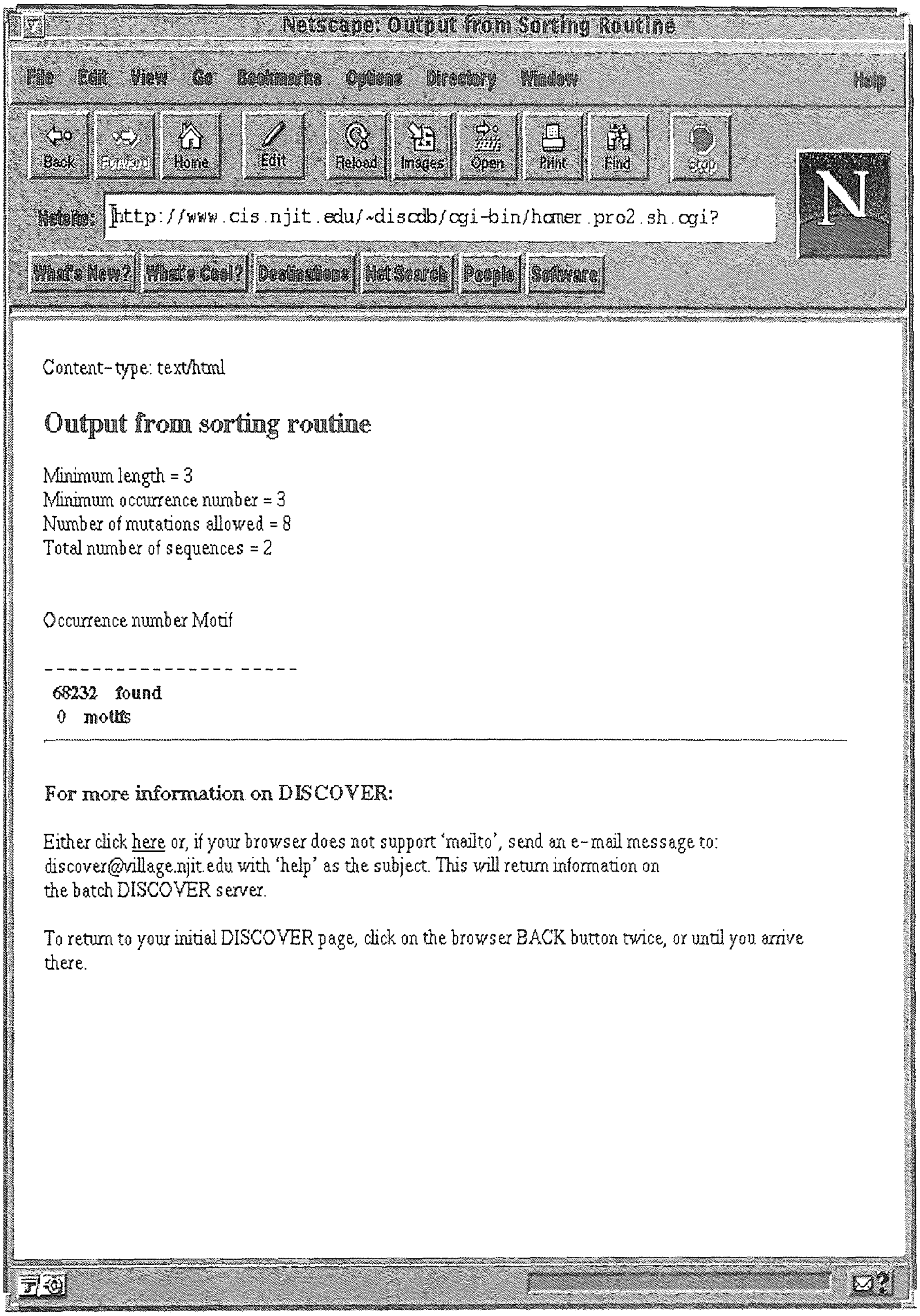
The quantity of allowed mutations: 8

To execute the Discovery Program, click on Execute

Execute

To return to input form. click on Browser Return Button

Figure A.3 Confirmation for SDISCOVER


Netscape: Output from Sorting Routine
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find Stop
Netsite: http://www.cis.njit.edu/~discdb/cgi-bin/homer.pro2.sh.cgi?
What's New? What's Cool? Destinations Net Search People Software

Content-type: text/html

Output from sorting routine

Minimum length = 3
Minimum occurrence number = 3
Number of mutations allowed = 8
Total number of sequences = 2

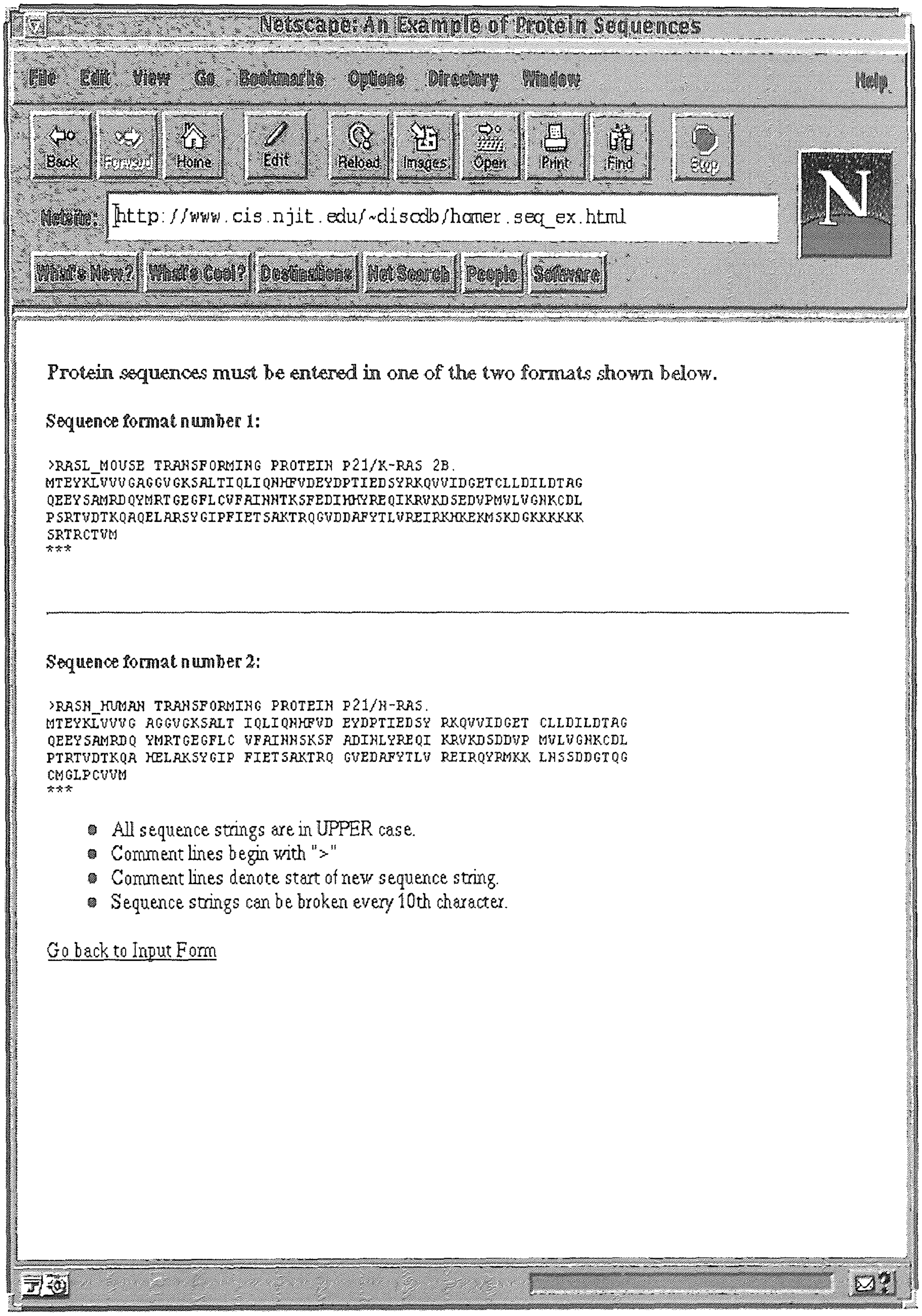
```
Occurrence number Motif

------------------ -----
 68232   found
  0   motifs
```

For more information on DISCOVER:

Either click here or, if your browser does not support 'mailto', send an e-mail message to: discover@village.njit.edu with 'help' as the subject. This will return information on the batch DISCOVER server.

To return to your initial DISCOVER page, click on the browser BACK button twice, or until you arrive there.

Figure A.4 Output From SDISCOVER


Netscape: An Example of Protein Sequences
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find Stop
Netsite: http://www.cis.njit.edu/~discdb/homer.seq_ex.html
What's New? What's Cool? Destinations Net Search People Software

Protein sequences must be entered in one of the two formats shown below.

Sequence format number 1:

```
>RASL_MOUSE TRANSFORMING PROTEIN P21/K-RAS 2B.
MTEYKLVVVGAGGVGKSALTIQLIQNHFVDEYDPTIEDSYRKQVVIDGETCLLDILDTAG
QEEYSAMRDQYMRTGEGFLCVFAINNTKSFEDIHHYREQIKRVKDSEDVPMVLVGNKCDL
PSRTVDTKQAQELARSYGIPFIETSAKTRQGVDDAFYTLVREIRKHKEKMSKDGKKKKKK
SRTRCTVM
***
```

Sequence format number 2:

```
>RASN_HUMAN TRANSFORMING PROTEIN P21/N-RAS.
MTEYKLVVVG AGGVGKSALT IQLIQNHFVD EYDPTIEDSY RKQVVIDGET CLLDILDTAG
QEEYSAMRDQ YMRTGEGFLC VFAINNSKSF ADINLYREQI KRVKDSDDVP MVLVGNKCDL
PTRTVDTKQA HELAKSYGIP FIETSAKTRQ GVEDAFYTLV REIRQYRMKK LNSSDDGTQG
CMGLPCVVM
***
```

- All sequence strings are in UPPER case.
- Comment lines begin with ">"
- Comment lines denote start of new sequence string.
- Sequence strings can be broken every 10th character.

Go back to Input Form

Figure A.5 Example of Protein Sequence Strings

A.4 TUTORIAL ON USING SDISCOVER ON WWW

Following is a tutorial on using SIDSCOVER on the World Wide Web.

1. Type in URL of SDISCOVER:

   `http://www.cis.njit.edu/~discdb/homer.pro4.html`

2. Type in your name, e-mail address, and your web site (if you have one)

3. Type in the following set of sequences:

```
>RASL_MOUSE TRANSFORMING PROTEIN P21/K-RAS 2B.
MTEYKLVVVGAGGVGKSALTIQLIQNHFVDEYDPTIEDSYRKQVVIDGETCLLDILDTAG
QEEYSAMRDQYMRTGEGFLCVFAINNTKSFEDIHHYREQIKRVKDSEDVPMVLVGNKCDL
PSRTVDTKQAQELARSYGIPFIETSAKTRQGVDDAFYTLVREIRKHKEKMSKDGKKKKKK
SRTRCTVM
>RASN_HUMAN TRANSFORMING PROTEIN P21/N-RAS.
MTEYKLVVVGAGGVGKSALTIQLIQNHFVDEYDPTIEDSYRKQVVIDGETCLLDILDTAG
QEEYSAMRDQYMRTGEGFLCVFAINNSKSFADINLYREQIKRVKDSDDVPMVLVGNKCDL
PTRTVDTKQAHELAKSYGIPFIETSAKTRQGVEDAFYTLVREIRQYRMKKLNSSDDGTQG
CMGLPCVVM
```

4. Choose the motif form of *X*

5. Click on 'Submit'

6. When the confirmation screen (see figure A.3, page 32) is shown, click on 'Execute' to submit the data to the SDISCOVER program.

APPENDIX B

RESOURCES FOR LaTeX FILE COMPARISON

B.1 INTRODUCTION

This appendix contains the following resources used in implementing TREEDIFF for LaTeX file comparison on the World Wide Web:

1. Perl and shell scripts

2. Screen shots of TREEDIFF for LaTeX

3. A tutorial on using TREEDIFF on WWW

B.2 SCRIPTS

This section contains the scripts[†] used to implement LaTeX file comparison on the World Wide Web. There are three scripts used:

1. `getlatex.cgi` - a Perl 5.003 script

2. `MOSAIC.FUNCTIONS` - a utility Bourne shell script, called by 'dolatex.sh.cgi'

3. `dolatex.sh.cgi` - a Bourne shell script

B.2.1 getlatex.cgi

Following is the Perl 5.003 script used to retrieve two LaTeX files via the URL's typed in by the user.

```
#!/home/discdb/local/bin/perl

use Config;
```

[†]N. B.: The text of this and the following scripts has been modified slightly. The modifications are primarily in the form of newlines inserted where they otherwise would not be. These modifications are imposed by the requirements of the thesis format.

```
# use URI::URL;
# use HTML::Parse;
# use HTTP::Response;
use HTTP::Status;
use LWP::Debug;
# use LWP::UserAgent;
use LWP::Simple;
use CGI;

$texfile1 = "tex1.dat";
$texfile2 = "tex2.dat";

my($q) = new CGI;

$| = 1; # flush headers now
# print $q->header('HTTP/1.0 200 OK');
print $q->header( -type => 'text/html',
                  -status => '200 OK',
                  );
print "\r\n";

print $q->start_html( -title=>'LaTeX File Retriever',
                      -author=>'pjohnson',
                      -BGCOLOR=>'#F0F0F0' );

print $q->h1('LaTex File Retriever');
print $q->p('&copy; NJIT DB-Lab, 1997');
print $q->hr;

$url1 = $q->param('texurl1');
$url2 = $q->param('texurl2');

$rc1 = getstore ($url1, $texfile1);

$rc2 = getstore ($url2, $texfile2);

# if return code eq not found for either or both, print out note
# stating
# that files must be observable by browser

if (is_success($rc1)) {
   print $q->h2('Successfully retrieved first LaTeX file!');
} else {
   print $q->h2('Error retrieving first LaTeX file!  Status: ',
                 status_message($rc1));
   if ($rc1 == RC_NOT_FOUND) {
# it'd be nice to show the url entered...
      print $q->h2('Could not find the document you specified.  The
                    document must be visible to the web browser.');
      print $q->h2('For example, if your web server requires all
                    accessible files to be placed under the
                    public_html directory,');
      print $q->h2('then the files you wish to analyze must be placed
                    there in a similar fashion.');
   }
}

if (is_success($rc2)) {
   print $q->h2('Successfully retrieved second LaTeX file!');
} else {
   print $q->h2('Error retrieving second LaTeX file!  Status: ',
                 status_message($rc2));
   if ($rc2 == RC_NOT_FOUND) {
# it'd be nice to show the url entered...
      print $q->h2('Could not find the document you specified.  The
                    document must be visible to the web browser.');
      print $q->h2('For example, if your web server requires all
                    accessible files to be placed under the
                    public_html directory,');
      print $q->h2('then the files you wish to analyze must be placed
                    there in a similar fashion.');
   }
}

# set up options for document comparison

$method = "post";
$action = "dolatex.sh.cgi";
%details = (
  "d", "show all categories, regardless of any changes in either
       file",
  "s", "show only those categories that have changes",
  "o", "default: show categories that have changes"
          );
%diffs = (
  "diff", "show output similar to Unix diff",
  "sdiff", "show output similar to Unix sdiff",
  "odiff", "show no difference output"
```

```
    );

if ((is_success($rc1)) && (is_success($rc2))) {
   print $q->startform($method, $action);

   print $q->h3('Having successfully retrieved both LaTeX
                 files, please choose the'+
                ' document comparison options');

   print $q->h3('Please select the level of detail to be shown on
                 output');
   print $q->radio_group(-name=>'detail',
                         -values=>['d','s','o'],
                         -default=>'o',
                         -linebreak=>'true',
                         -labels=>\%details);

   print hr;

   print $q->h3('Please choose the difference reporting options');
   print $q->radio_group(-name=>'difference',
                         -values=>['diff','sdiff','odiff'],
                         -default=>'odiff',
                         -linebreak=>'true',
                         -labels=>\%diffs);
   print $q->hr;

   print $q->submit('Begin document comparison');
   print $q->defaults('Reset');
   print $q->endform;
}

print $q->end_html;

exit 0;
```

B.2.2 MOSAIC.FUNCTIONS

Following is the utility file entitled, "MOSAIC.FUNCTIONS". This is called by 'dolatex.sh.cgi'.

```
#
# Translate escapes from the form to real characters.
```

```
#

# note mods:  the single quote %27 is eliminated ('); %0D = <BR>
unescape_url ()
{
        echo "${@}" | tr '+' ' ' | tr -d '
' |
        sed -e 's/%07/^G/g' -e 's/%08/^H/g'
            -e 's/%09/  /g' -e 's/%0A/\
/g'\
            -e 's/%21/!/g'  -e 's/%22/"/g'  \
            -e 's/%23/#/g'  -e 's/%24/$/g'  \
            -e 's/%25/\%/g' -e 's/%26/\&/g' \
            -e "s/%27//g"   -e 's/%28/(/g'  \
            -e 's/%29/)/g'  -e 's/%2B/+/g'  \
            -e 's/%2C/,/g'  -e 's/%2F/\//g' \
            -e 's/%3A/:/g'  -e 's/%3B/;/g'  \
            -e 's/%3C/</g'  -e 's/%3D/=/g'  \
            -e 's/%3E/>/g'  -e 's/%3F/?/g'  \
            -e 's/%5B/[/g'  -e 's/%5C/\\/g' \
            -e 's/%5D/]/g'  -e 's/%5E/^/g'  \
            -e 's/%60/`/g'  -e 's/%7B/{/g'  \
            -e 's/%7C/|/g'  -e 's/%7D/}/g'  \
            -e 's/%7E/~/g'  -e 's/%0D/<BR>/g'
}

#
# Read the name-value pairs from stdin (from a POSt form), and
# put them into the environment.
#

read_form()
{
        read line
        line=`echo "${line}" | tr -d '\015'`

        while [ "${line}" ]
        do
                work=`echo "${line}" | cut -d'&' -f1`
                line=`echo "${line}" | cut -d'&' -f2-`

                name=`echo "${work}" | cut -d= -f1`
                value=`echo "${work}" | cut -d= -f2-`
                if [ "${value}" ]
                then
                        value=`unescape_url "${value}" `
                fi

                eval "${name}='${value}'"

                if [ "${work}" = "${line}" ]
                then
                        break
                fi
        done
}
```

B.2.3 dolatex.sh.cgi

Following is the Bourne shell script used to process the LaTeX file comparison options. This shell script also executes the file comparison program.

```
#!/bin/sh

#####
#
# this code from Padmaja Balabhadrapatruni
# written for CS project under Dr. J Wang
#
# written to run on Oak, ported to homer by Phil Johnson 4-29-97
#
#####

. ./MOSAIC.FUNCTIONS

#########################################################

cat <<END
Content-type: text/html

<TITLE>LaTeX File Comparison (execution)</TITLE>

<h2>File Comparison System Response</h2>
<P>
END
```

```
##########################################################

#Take the single line of input that is passed from the form ...

echo "<h3>about to read line from stdin</h3>"
read line
line=`echo "${line}" | tr -d '\015'`
echo ${line} > /home/discdb/test/testfile
newline=`unescape_url ${line}`
echo ${newline} > /home/discdb/test/newtest

        # This section processes the input line and extracts the
        # variables one
        # at a time and stores them in variables.

        # Extracting the first item. If data is not entered in
        # a certain text box, an error message is displayed, and
        # the program
        # exits. Until all the required data is not entered,
        # the program will not proceed further.

        # get the detail spec... (radio button)
        #
        work=`echo "${newline}" | cut -d'&' -f1`
        newline=`echo "${newline}" | cut -d'&' -f2-`
        name=`echo "${work}" | cut -d= -f1`
        value=`echo "${work}" | cut -d= -f2-`
        if [ "$value" ]
        then
                det="$value"
        else
                echo "<B> ERROR! PLEASE ENTER ALL THE DATA!!! (detail
                      spec) </B><P>"
        exit 1
        fi

        echo "<br> the value of the detail spec is: <b> $det </b>"

        # get the difference spec... (radio button)
        #
        work=`echo "${newline}" | cut -d'&' -f1`
        newline=`echo "${newline}" | cut -d'&' -f2-`
        name=`echo "${work}" | cut -d= -f1`
        value=`echo "${work}" | cut -d= -f2-`
        if [ "$value" ]
        then
                diff="$value"
        else
                echo "<B> ERROR! PLEASE ENTER ALL THE DATA!!!
                      (difference spec) </B><P>"
        exit 1
        fi

        echo "<br> the value of the difference spec is:
              <b> $diff </b>"

        echo "<br> all done extracting data "

        # set up options...

        case "$det" in
                o)det1=;;
                d)det1='-d';;
                s)det1='-s';;
        esac

        case "$diff" in
                odiff)diff1=;;
                diff)diff1='-diff';;
                sdiff)diff1='-sdiff';;
        esac

        echo "<br> value of diff1 is <b> $diff1 </b>"
        echo "<br> value of det1 is <b> $det1 </b>"

# now call tdlatex with command-line parameters...

echo "<pre>"
/home/discdb/public_html/cgi-bin/tdlatex $det1 $diff1 tex1.dat \
tex2.dat
echo "</pre>"
#*********************      E N D    ***********************
```

B.3 SCREEN SHOTS

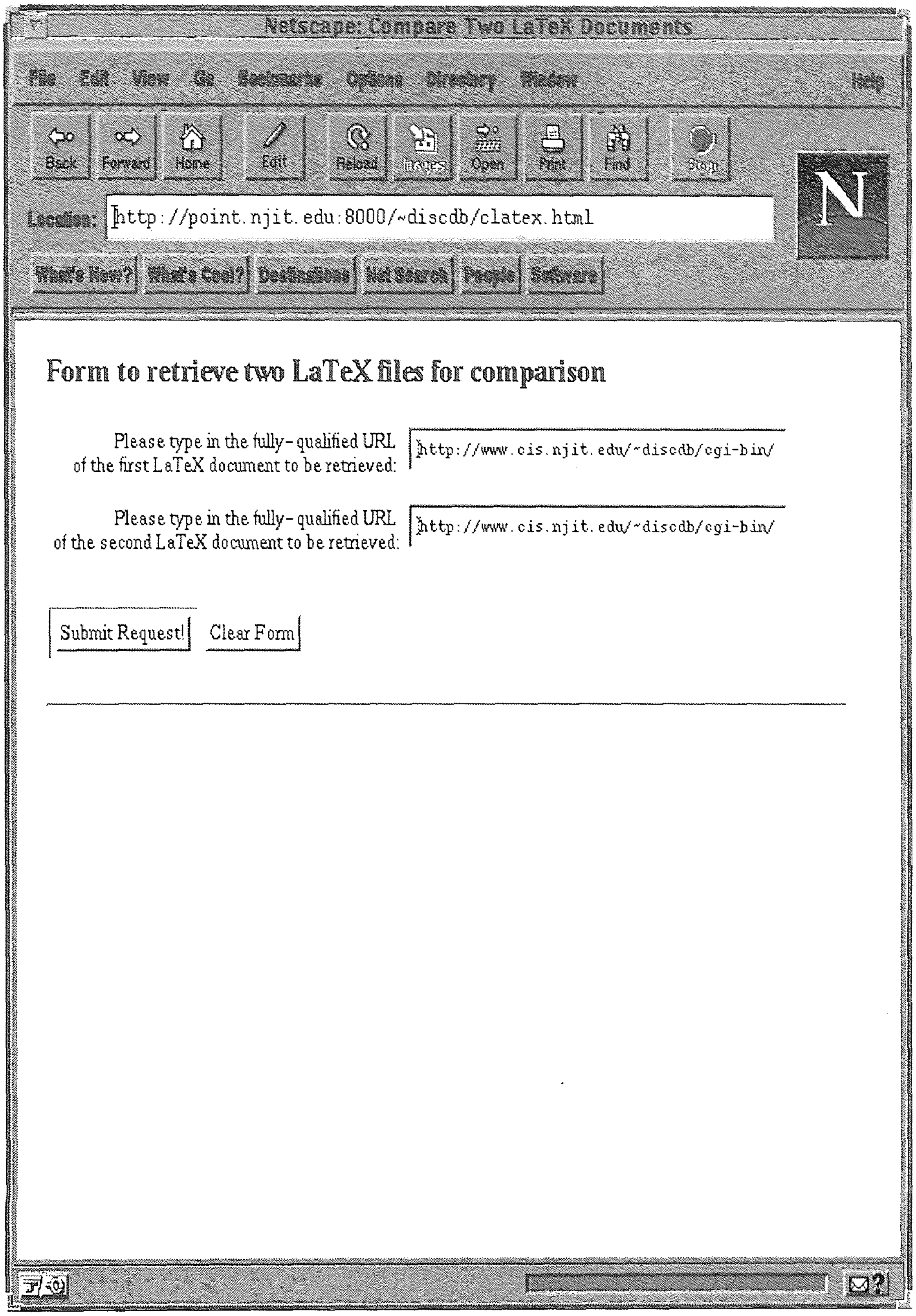
Following are the screen shots of TREEDIFF for LaTeX file comparison on the WWW. There are four shots:

1. LaTeX File Retrieval, Fig. B.1

2. Notification of Successful Retrieval, Fig. B.2

3. Output of LaTeX File Comparison, Fig. B.3

4. Notification of a LaTeX File Not Found, Fig. B.4



http://point.njit.edu:8000/~discdb/clatex.html

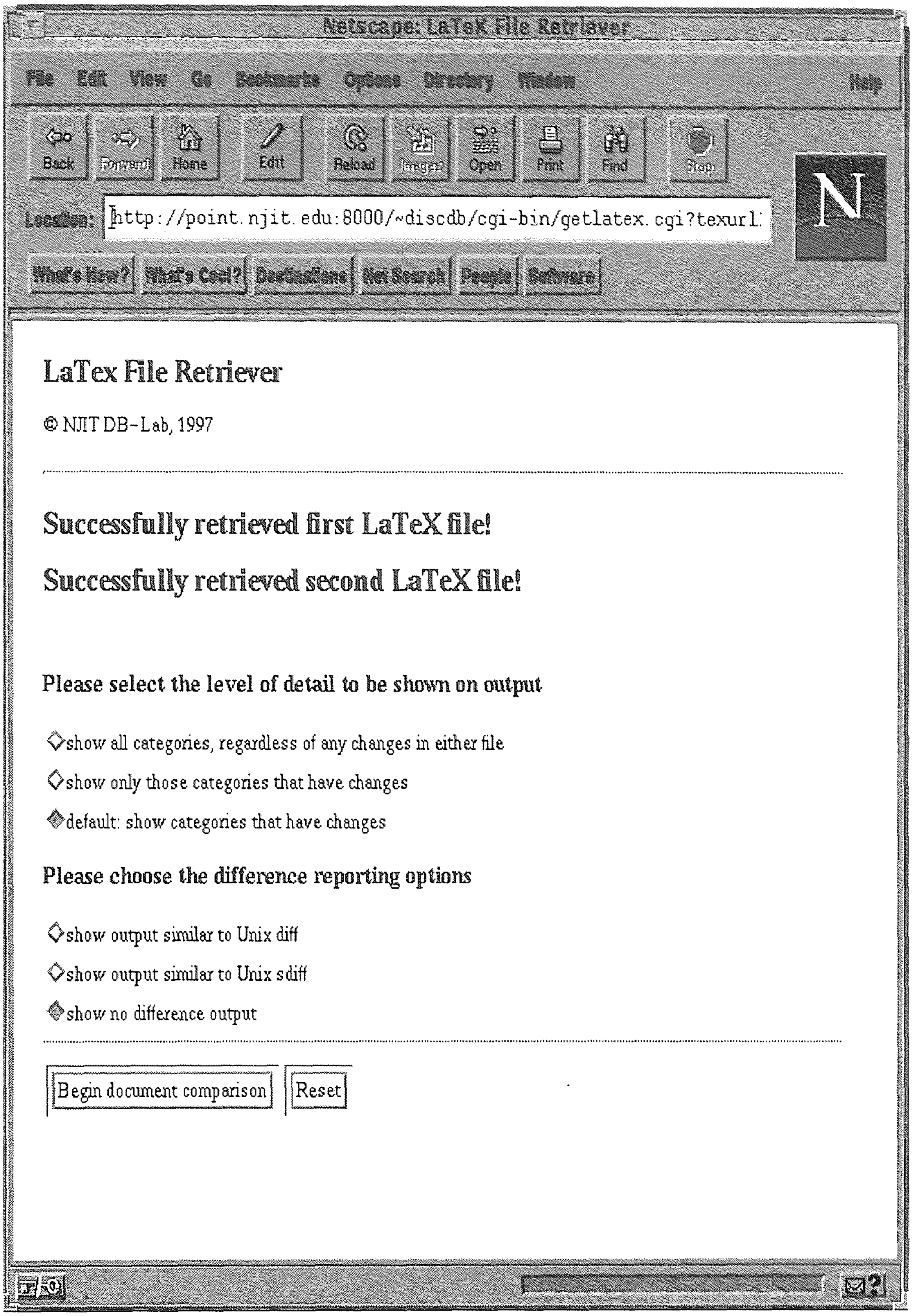
Form to retrieve two LaTeX files for comparison

Please type in the fully-qualified URL of the first LaTeX document to be retrieved: http://www.cis.njit.edu/~discdb/cgi-bin/

Please type in the fully-qualified URL of the second LaTeX document to be retrieved: http://www.cis.njit.edu/~discdb/cgi-bin/

Submit Request! Clear Form

Figure B.1 LaTeX File Retrieval


Netscape: LaTeX File Retriever
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find Stop
Location:
What's New? What's Cool? Destinations Net Search People Software

http://point.njit.edu:8000/~discdb/cgi-bin/getlatex.cgi?texurl:

LaTex File Retriever

© NJIT DB-Lab, 1997

Successfully retrieved first LaTeX file!

Successfully retrieved second LaTeX file!

Please select the level of detail to be shown on output

◇ show all categories, regardless of any changes in either file

◇ show only those categories that have changes

[✓ ◆ default: show categories that have changes]

Please choose the difference reporting options

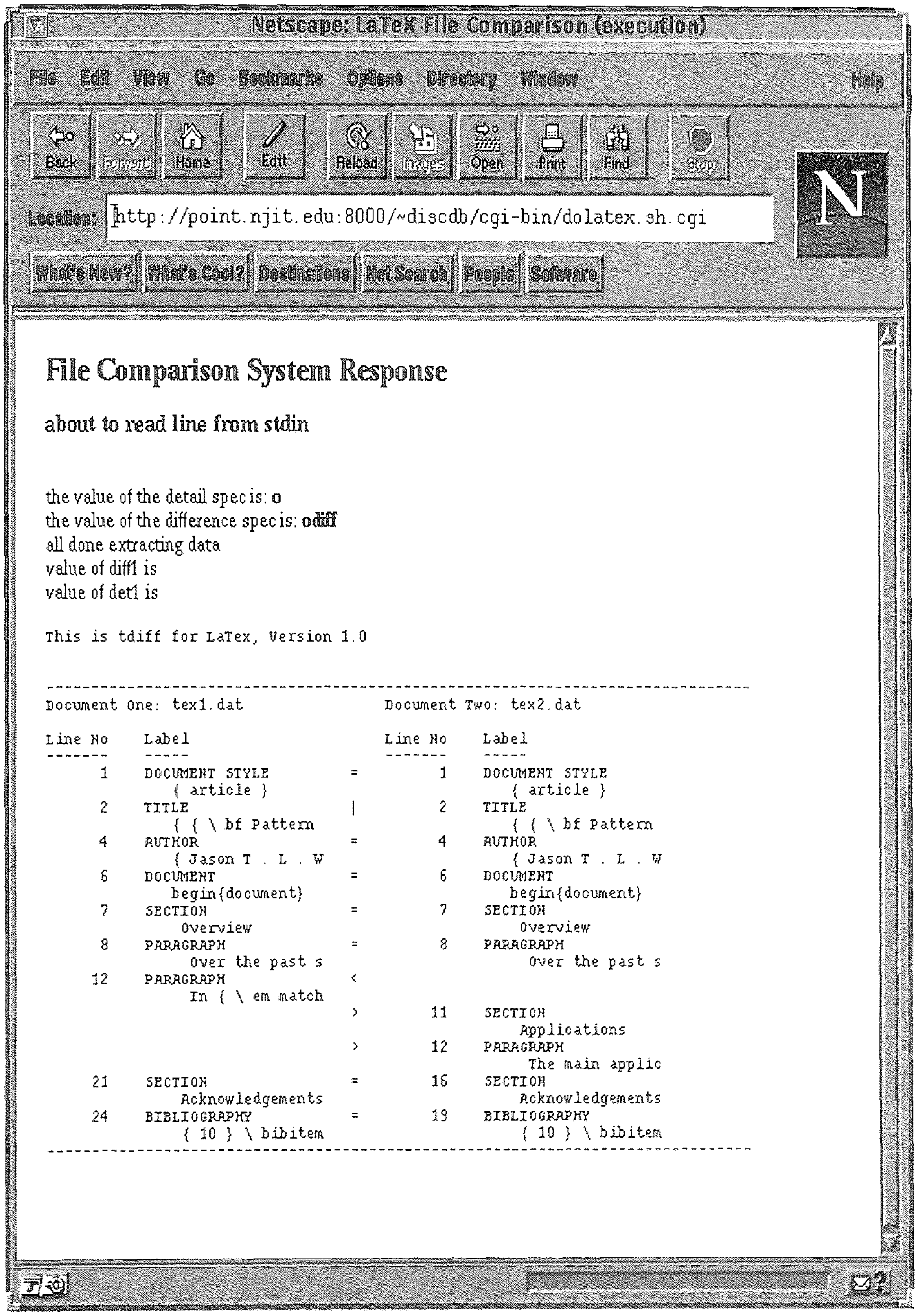
◇ show output similar to Unix diff

◇ show output similar to Unix sdiff

[✓ ◆ show no difference output]

Begin document comparison | Reset

Figure B.2 Notification of Successful Retrieval of Two LaTeX Documents


Netscape: LaTeX File Comparison (execution)
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find Stop
Location: http://point.njit.edu:8000/~discdb/cgi-bin/dolatex.sh.cgi
What's New? What's Cool? Destinations Net Search People Software

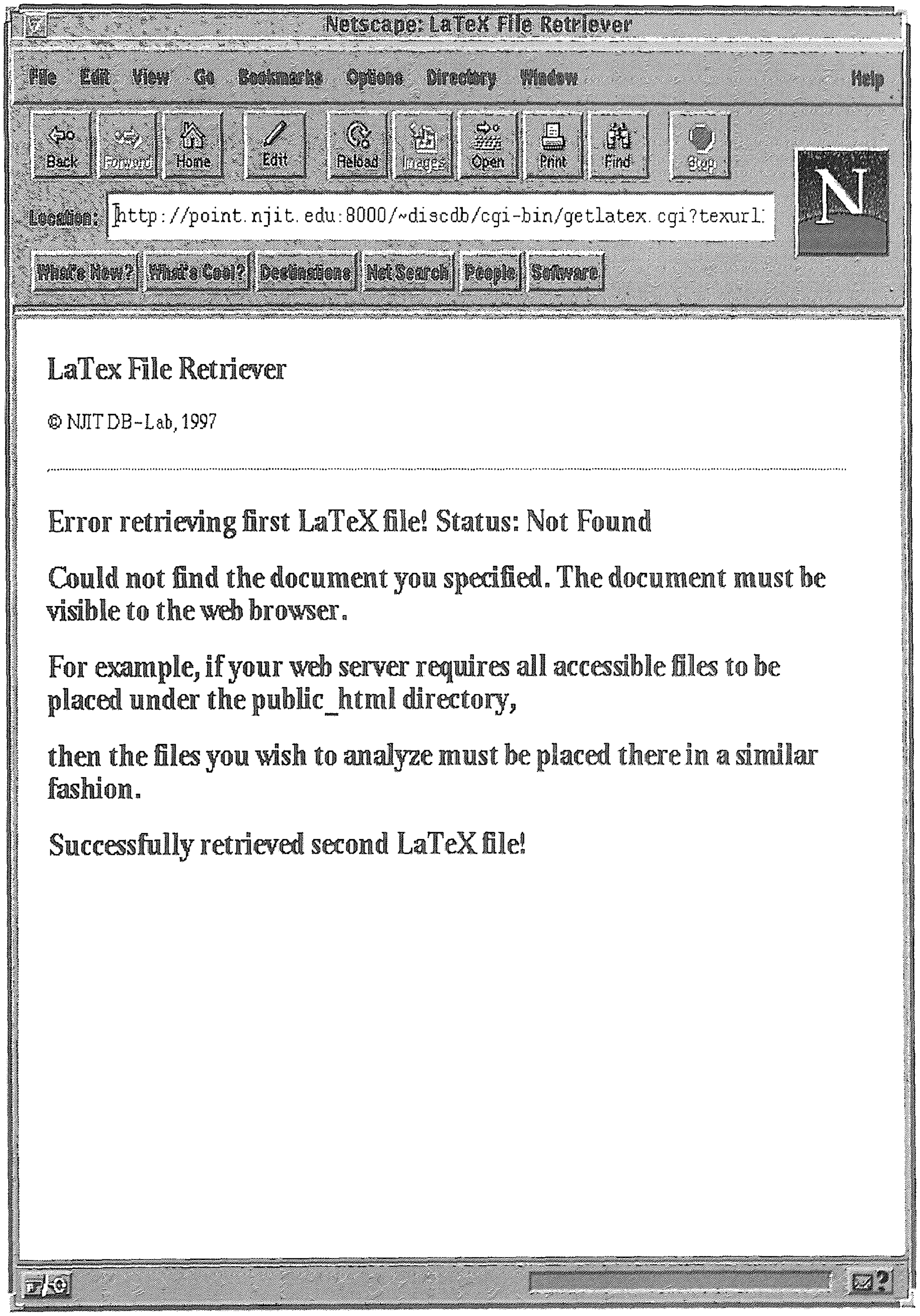
File Comparison System Response

about to read line from stdin

the value of the detail spec is: **o**
the value of the difference spec is: **odiff**
all done extracting data
value of diff1 is
value of det1 is

```
This is tdiff for LaTex, Version 1.0


----------------------------------------------------------------------------
Document One: tex1.dat                Document Two: tex2.dat

Line No    Label                      Line No    Label
-------    -----                      -------    -----
      1    DOCUMENT STYLE        =          1    DOCUMENT STYLE
              { article }                           { article }
      2    TITLE                 |          2    TITLE
              { { \ bf Pattern                      { { \ bf Pattern
      4    AUTHOR                =          4    AUTHOR
              { Jason T . L . W                     { Jason T . L . W
      6    DOCUMENT              =          6    DOCUMENT
              begin{document}                       begin{document}
      7    SECTION               =          7    SECTION
               Overview                              Overview
      8    PARAGRAPH             =          8    PARAGRAPH
                Over the past s                       Over the past s
     12    PARAGRAPH             <
                In { \ em match
                                 >         11    SECTION
                                                     Applications
                                 >         12    PARAGRAPH
                                                      The main applic
     21    SECTION               =         16    SECTION
               Acknowledgements                      Acknowledgements
     24    BIBLIOGRAPHY          =         19    BIBLIOGRAPHY
               { 10 } \ bibitem                      { 10 } \ bibitem
----------------------------------------------------------------------------
```

Figure B.3 Output of LaTeX File Comparison


Netscape: LaTeX File Retriever
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find Stop
Location:
What's New? What's Cool? Destinations Net Search People Software

```
http://point.njit.edu:8000/~discdb/cgi-bin/getlatex.cgi?texurl:
```

LaTex File Retriever

© NJIT DB-Lab, 1997

Error retrieving first LaTeX file! Status: Not Found

Could not find the document you specified. The document must be visible to the web browser.

For example, if your web server requires all accessible files to be placed under the public_html directory,

then the files you wish to analyze must be placed there in a similar fashion.

Successfully retrieved second LaTeX file!

Figure B.4 Notification of a LaTeX File Not Found

B.4 TUTORIAL ON USING TREEDIFF FOR LaTeX ON WWW

Following is a tutorial on using TREEDIFF for LaTeX on the World Wide Web.

1. Access the TREEDIFF for LaTeX web page at:

   `http://www.cis.njit.edu/~discdb/clatex.html`

2. Type in the following URLs for the LaTeX documents:

   `http://www.cis.njit.edu/~discdb/cgi-bin/TDLATEX_SRC/doc1.tex`

   `http://www.cis.njit.edu/~discdb/cgi-bin/TDLATEX_SRC/doc2.tex`

3. When the web page indicating successful retrieval s displayed (see figure B.2, page 46), choose (accept) the default document comparison options and click on 'Begin document comparison'

4. The results will be shown on the next (web) page (see figure B.3, page 47).

APPENDIX C

RESOURCES FOR SGML FILE COMPARISON

C.1 INTRODUCTION

This appendix contains the following resources used in implementing SGML file comparison on the World Wide Web:

1. Perl and shell scripts
2. Screen shots
3. A tutorial on using TREEDIFF for SGML file comparison

C.2 SCRIPTS

This section contains the scripts[†] used in performing the SGML file comparison. There are three scripts used for SGML file comparison:

1. `getsgml.cgi` - a Perl 5.003 script
2. `MOSAIC.FUNCTIONS` - a utility Bourne shell script, called by 'dosgml.sh.cgi'
3. `dosgml.sh.cgi` - a Bourne shell script

C.2.1 getsgml.cgi

Following is the Perl 5.003 script used to retrieve two SGML files via the URLs typed in by the user.

```
#!/home/discdb/local/bin/perl

#####
```

[†]N. B.: The text of this and the following scripts has been modified slightly. The modifications are primarily in the form of newlines inserted where they otherwise would not be. These modifications are imposed by the requirements of the thesis format.

```
#
# file: getsgml.cgi
#
# date written: 6-10-97
# author:       Phil Johnson
# who:          RA for Dr. J. T. L. Wang
#
# purpose: retrieve two sgml files via URL's entered by user on web
#          page
#
#####

use Config;
use HTTP::Status;
use LWP::Debug;
use LWP::Simple;
use CGI;

$texfile1 = "sgml1.dat";
$texfile2 = "sgml2.dat";

my($q) = new CGI;

$| = 1; # flush headers now
# print $q->header('HTTP/1.0 200 OK');
print $q->header( -type => 'text/html',
                  -status => '200 OK',
                  );
print "\r\n";

print $q->start_html( -title=>'SGML File Retriever',
                      -author=>'pjohnson',
                      -BGCOLOR=>'#F0F0F0' );

print $q->h1('SGML File Retriever');
print $q->p('&copy; NJIT DB-Lab, 1997');
print $q->hr;

$url1 = $q->param('texurl1');
$url2 = $q->param('texurl2');

$rc1 = getstore ($url1, $texfile1);

$rc2 = getstore ($url2, $texfile2);
```

```
# if return code eq not found for either or both, print out note
# stating
# that files must be observable by browser

if (is_success($rc1)) {
   print $q->h2('Successfully retrieved first SGML file!');
} else {
   print $q->h2('Error retrieving first SGML file!  Status: ',
                status_message($rc1));
   if ($rc1 == RC_NOT_FOUND) {
# it'd be nice to show the url entered...
      print $q->h3('Could not find the first document you specified.
                    The document must be visible to the web
                    browser.');
      print $q->h3('For example, if your web server requires all
                    accessible files to be placed under the
                    public_html directory,');
      print $q->h3('then the files you wish to analyze must be
                    placed there in a similar fashion.');
   }
}

if (is_success($rc2)) {
   print $q->h2('Successfully retrieved second SGML file!');
} else {
   print $q->h2('Error retrieving second SGML file!  Status: ',
                status_message($rc2));
   if ($rc2 == RC_NOT_FOUND) {
# it'd be nice to show the url entered...
      print $q->h3('Could not find the second document you specified.
                    The document must be visible to the web
                    browser.');
      print $q->h3('For example, if your web server requires all
                    accessible files to be placed under the
                    public_html directory,');
      print $q->h3('then the files you wish to analyze must be placed
                    there in a similar fashion.');
   }
}

# set up options for document comparison

$method = "post";
```

```
$action = "dosgml.sh.cgi";  # this cgi script will process
                            # & execute...

%details = (
     "d", "show all categories, regardless of any changes in either
           file",
     "s", "show only those categories that have changes",
     "o", "default: show categories that have changes"
           );

%diffs = (
  "diff", "show output similar to Unix diff",
  "sdiff", "show output similar to Unix sdiff",
  "odiff", "show no difference output"
    );

if ((is_success($rc1)) && (is_success($rc2))) {
   print $q->startform($method, $action);

   print $q->h3('Having successfully retrieved both SGML
                files, please
                choose the document comparison options');

   print $q->h3('Please select the level of detail to be shown on
                output');
   print $q->radio_group(-name=>'detail',
                        -values=>['d','s','o'],
                        -default=>'o',
                        -linebreak=>'true',
                        -labels=>\%details);

   print $q->hr;

   print $q->h3('Please choose the difference reporting options');
   print $q->radio_group(-name=>'difference',
                        -values=>['diff','sdiff','odiff'],
                        -default=>'odiff',
                        -linebreak=>'true',
                        -labels=>\%diffs);
   print $q->hr;

   print $q->submit('Begin document comparison');
   print $q->defaults('Reset');
   print $q->endform;
}

print $q->end_html;

exit 0;
```

C.2.2 MOSAIC.FUNCTIONS

Following is the utility file entitled, "MOSAIC.FUNCTIONS". This is called by 'dosgml.sh.cgi'.

```
#
# Translate escapes from the form to real characters.
#

# note mods:  the single quote %27 is eliminated ('); %0D = <BR>
unescape_url ()
{
        echo "${@}" | tr '+' ' ' | tr -d ' ' |
        sed -e 's/%07/^G/g' -e 's/%08/^H/g'
            -e 's/%09/  /g' -e 's/%0A/\
/g'\
            -e 's/%21/!/g'  -e 's/%22/"/g'  \
            -e 's/%23/#/g'  -e 's/%24/$/g'  \
            --e 's/%25/\%/g' -e 's/%26/\&/g' \
            -e "s/%27//g"   -e 's/%28/(/g'  \
            -e 's/%29/)/g'  -e 's/%2B/+/g'  \
            -e 's/%2C/,/g'  -e 's/%2F/\//g' \
            -e 's/%3A/:/g'  -e 's/%3B/;/g'  \
            -e 's/%3C/</g'  -e 's/%3D/=/g'  \
            -e 's/%3E/>/g'  -e 's/%3F/?/g'  \
            -e 's/%5B/[/g'  -e 's/%5C/\\/g' \
            -e 's/%5D/]/g'  -e 's/%5E/^/g'  \
            -e 's/%60/`/g'  -e 's/%7B/{/g'  \
            -e 's/%7C/|/g'  -e 's/%7D/}/g'  \
            -e 's/%7E/~/g'  -e 's/%0D/<BR>/g'
}


#
# Read the name-value pairs from stdin (from a POSt form), and
# put them into the environment.
```

```
#

read_form()
{
        read line
        line=`echo "${line}" | tr -d '\015'`

        while [ "${line}" ]
        do
                work=`echo "${line}" | cut -d'&' -f1`
                line=`echo "${line}" | cut -d'&' -f2-`

                name=`echo "${work}" | cut -d= -f1`
                value=`echo "${work}" | cut -d= -f2-`

                if [ "${value}" ]
                then
                        value=`unescape_url "${value}" `
                fi

                eval "${name}='${value}'"

                if [ "${work}" = "${line}" ]
                then
                        break
                fi
        done
}
```

C.2.3 dosgml.sh.cgi

Following is the Bourne shell script used to process the file comparison options specifed by the user. This script also executes the SGML comparison program.

```
#!/bin/sh

#####
#
# this code from Padmaja Balabhadrapatruni
# written for CS project under Dr. J Wang
#
# written to run on Oak, ported to homer by Phil Johnson 4-29-97
```

```
#
#####

. ./MOSAIC.FUNCTIONS

#
# Trap hangup, interrupt, and quit signals; exit gracefully
# This shouldn't be an issue, but is included to be safe.
#

trap  'echo "Content-type: text/plain\n\n";
         echo "Your request could not be processed.\n";
         echo "An internal error occurred ( an interrupt was
               caught ).";
         echo "Please try your request again.  If the problem";
         echo "persists, please contact the webmaster.";
         exit 1'       1 2 3

##########################################################

cat <<END
Content-type: text/html

<TITLE>SGML File Comparison (execution)</TITLE>

<h2>File Comparison System Response</h2>
<P>
END

##########################################################

#Take the single line of input that is passed from the form ...

        echo "<h3>about to read line from stdin</h3>"
        read line
        line=`echo "${line}" | tr -d '\015'`
        echo ${line} > /home/discdb/test/testfile
        newline=`unescape_url ${line}`
        echo ${newline} > /home/discdb/test/newtest

# This section processes the input line and extracts the
# variables one
# at a time and stores them in variables.
# Extracting the first item. If data is not entered in
# a certain text box, an error message is displayed, and
# the program
# exits. Until all the required data is not entered,
# the program will not proceed further.

# get the detail spec... (radio button)
#
        work=`echo "${newline}" | cut -d'&' -f1`
        newline=`echo "${newline}" | cut -d'&' -f2-`
        name=`echo "${work}" | cut -d= -f1`
        value=`echo "${work}" | cut -d= -f2-`
        if [ "$value" ]
        then
                det="$value"
        else
                echo "<B> ERROR! PLEASE ENTER ALL THE DATA!!! (detail
                      spec) </B><P>"
        exit 1
        fi

        echo "<br> the value of the detail spec is: <b> $det </b>"

# get the difference spec... (radio button)
#
        work=`echo "${newline}" | cut -d'&' -f1`
        newline=`echo "${newline}" | cut -d'&' -f2-`
        name=`echo "${work}" | cut -d= -f1`
        value=`echo "${work}" | cut -d= -f2-`
        if [ "$value" ]
        then
                diff="$value"
        else
                echo "<B> ERROR! PLEASE ENTER ALL THE DATA!!!
                      (difference spec) </B><P>"
        exit 1
        fi

        echo "<br> the value of the difference spec is:
              <b> $diff </b>"

        echo "<br> all done extracting data "

# set up options...
```

```
        case "$det" in
                o)det1=;;
                d)det1='-d';;
                s)det1='-s';;
        esac

        case "$diff" in
                odiff)diff1=;;
                diff)diff1='-diff';;
                sdiff)diff1='-sdiff';;
        esac

        echo "<br> value of diff1 is <b> $diff1 </b>"
        echo "<br> value of det1 is <b> $det1 </b>"

echo "<br> about to call tdsgml via sgml.sh (modified
      invokeDCT.sh...)<br>"

# now call tdlatex with command-line parameters...

#
# File: sgml.sh
#
# June 1997, Phil Johnson
# modified from: invokeDCT.sh, Jan. 1996, Brophy
#
# This shell script supports the CGI processing related to the HTML
# form which allows users to compare 2 document revisions using
# George's tdsgml
#

umask 002

#
# Create a temporary directory for the document comparison
#

DCTTmpDir=/tmp/DCT$$

/bin/rm -fr ${DCTTmpDir} 2> /dev/null

/bin/mkdir ${DCTTmpDir} 2>/dev/null
```

```
# check to see if command for directory creation succeeded...
# most recent status is in $?...
# status equal to zero implies successful completion...

if [ $? -ne 0 ]
then

        # print the MIME header and an error message, and quit

        # echo "Content-type: text/plain\n\n";
        echo "Your request could not be processed.\n"
        echo "Error creating temporary directory for processing
              output from tdsgml.\n"
        echo "Please try your request again.  If the problem"
        echo "persists, please contact the webmaster."
        exit 1
fi

# set directory where tdsgml files (executables & such) live ...

H=/home/discdb/public_html/cgi-bin/TDSGML_SRC

cp sgml1.dat $H/sgml1.dat

# check for successful file copy...

if [ $? -ne 0 ]
then

        # print the MIME header and an error message, and quit

        # echo "Content-type: text/plain\n\n";
        echo "Your request could not be processed.\n"
        echo "Error copying temporary file to working directory.\n"
        echo "Please try your request again.  If the problem"
        echo "persists, please contact the webmaster."
        exit 1
fi

cp sgml2.dat $H/sgml2.dat

# check for successful file copy...

if [ $? -ne 0 ]
```

```
then

        # print the MIME header and an error message, and quit

        # echo "Content-type: text/plain\n\n";
        echo "Your request could not be processed.\n"
        echo "Error copying temporary file to working directory.\n"
        echo "Please try your request again.  If the problem"
        echo "persists, please contact the webmaster."
        exit 1
fi

#
# now run tdsgml
#

# cd ${DCTTmpDir}

$H/tdsgml $diff1 $det1 $H/catalog $H/html.decl $H/sgml1.dat
          $H/sgml2.dat \
        > ${DCTTmpDir}/tdsgml.out 2> ${DCTTmpDir}/tdsgml.err

#
# The tdsgml program always exits with 0 status, even on an error
# condition.  If the error file is empty, we conclude the program
# ran successfully.
#

if [ ! -s ${DCTTmpDir}/tdsgml.err ]
then
        # successful comparison run:

        # print the MIME header and the comparison results, and quit

        # echo "Content-type: text/plain\n\n";
        echo "<pre>"
        cat ${DCTTmpDir}/tdsgml.out
        echo "</pre>"

        #
        # no longer need the tmp directory or files; remove them
        #

        /bin/rm -fr ${DCTTmpDir} 2> /dev/null
        exit 0

else
        # something went wrong...

        # print the MIME header and an error message, and quit

        # echo "Content-type: text/plain\n\n";
        echo "Your request could not be processed.\n"
        echo "An error occurred while comparing the documents.\n"
        echo "Please contact the webmaster regarding this error.\n"

        #
        # leave the DCTTmpDir directory there for problem
        # investigation
        #

exit 1
fi

# end ...
```

C.3 SCREEN SHOTS

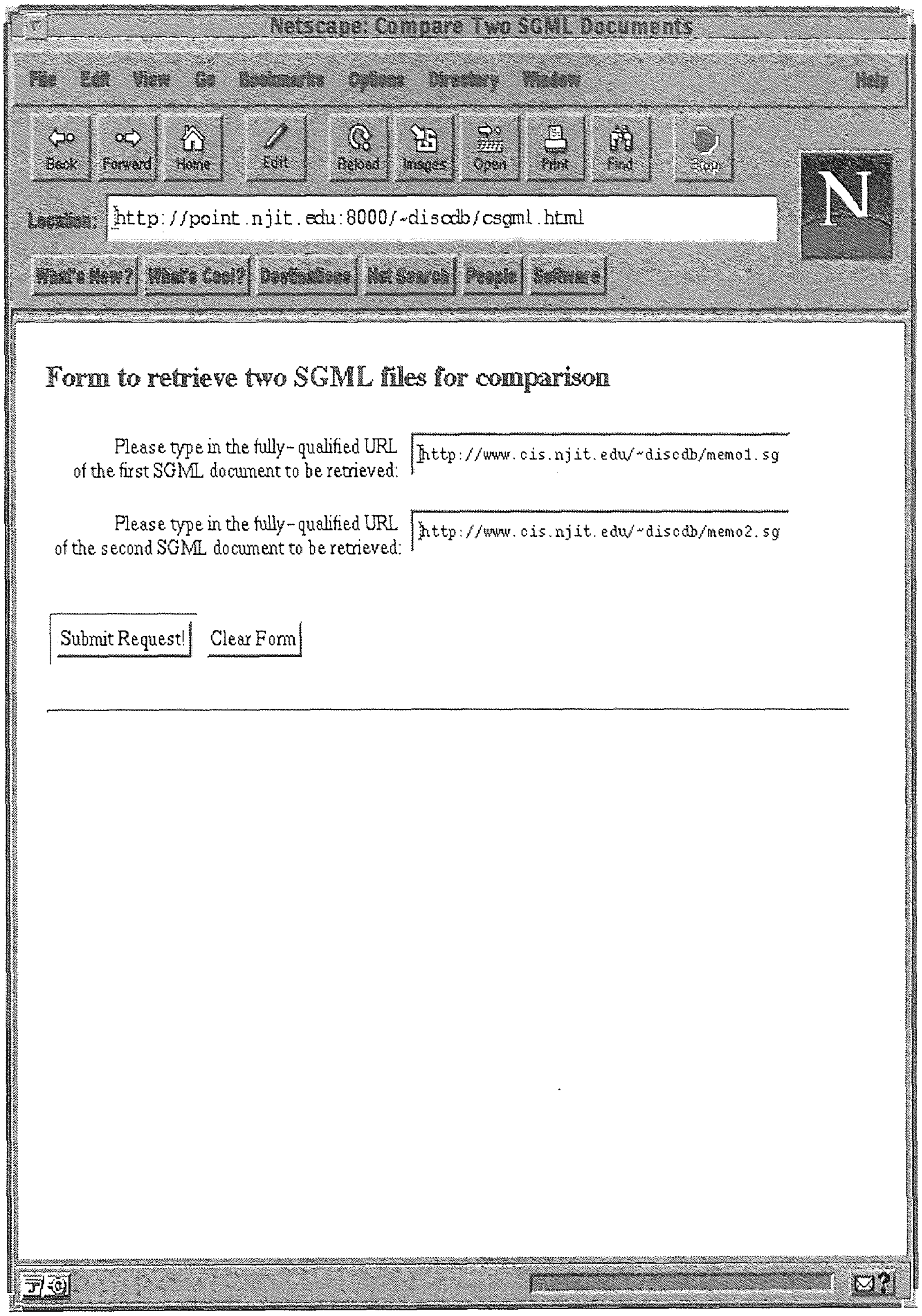
Following are the screen shots of TREEDIFF for SGML file comparison on the WWW. There are four shots:

1. SGML File Retrieval, Fig. C.1

2. Notification of Successful Retrieval, Fig. C.2

3. Output of SGML File Comparison, Fig. C.3

4. Notification of an SGML File Not Found, Fig. C.4


Netscape: Compare Two SGML Documents
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find Stop
What's New? What's Cool? Destinations Net Search People Software

Location: http://point.njit.edu:8000/~discdb/csgml.html

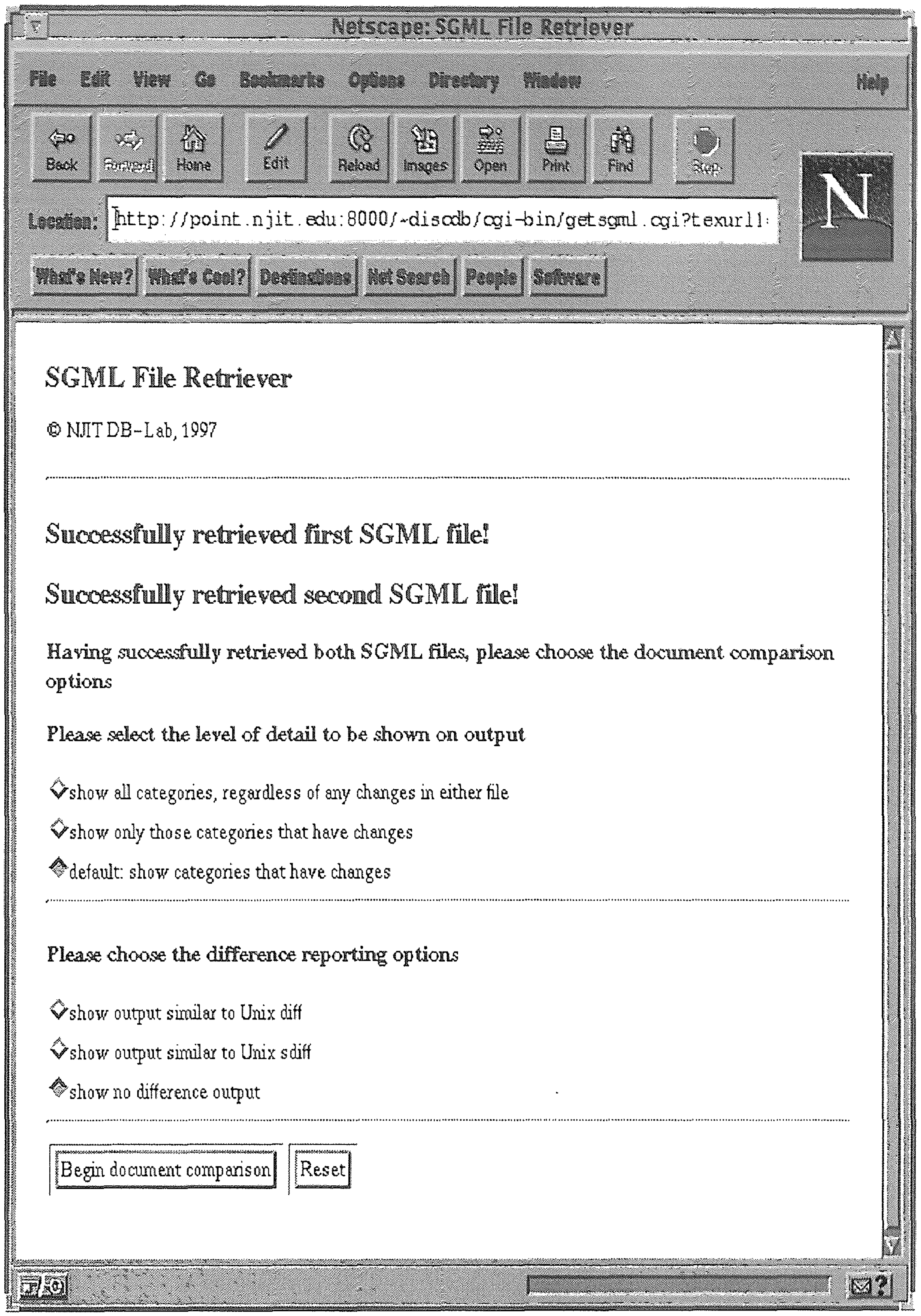
Form to retrieve two SGML files for comparison

Please type in the fully-qualified URL of the first SGML document to be retrieved: http://www.cis.njit.edu/~discdb/memo1.sg

Please type in the fully-qualified URL of the second SGML document to be retrieved: http://www.cis.njit.edu/~discdb/memo2.sg

Submit Request! Clear Form

Figure C.1 SGML File Retrieval


Netscape: SGML File Retriever
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find Stop
Location: http://point.njit.edu:8000/~discdb/cgi-bin/getsgml.cgi?texurl1=
What's New? What's Cool? Destinations Net Search People Software

SGML File Retriever

© NJIT DB-Lab, 1997

Successfully retrieved first SGML file!

Successfully retrieved second SGML file!

Having successfully retrieved both SGML files, please choose the document comparison options

Please select the level of detail to be shown on output

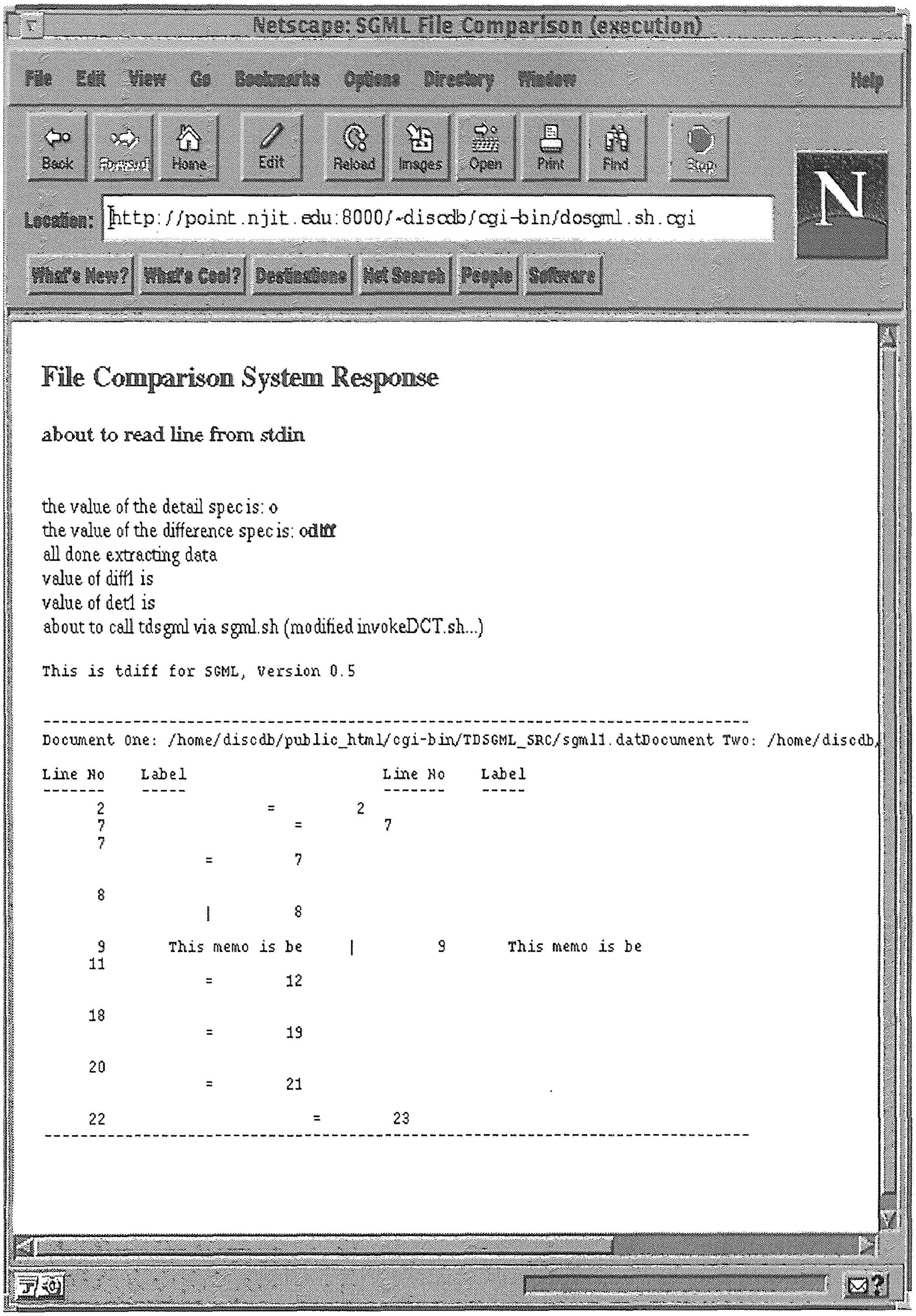
- show all categories, regardless of any changes in either file
- show only those categories that have changes
- [✓ default: show categories that have changes]

Please choose the difference reporting options

- show output similar to Unix diff
- show output similar to Unix sdiff
- [✓ show no difference output]

Begin document comparison | Reset

Figure C.2 Notification of Successful Retrieval of Two SGML Files


Netscape: SGML File Comparison (execution)
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find Stop
Location: http://point.njit.edu:8000/~discdb/cgi-bin/dosgml.sh.cgi
What's New? What's Cool? Destinations Net Search People Software

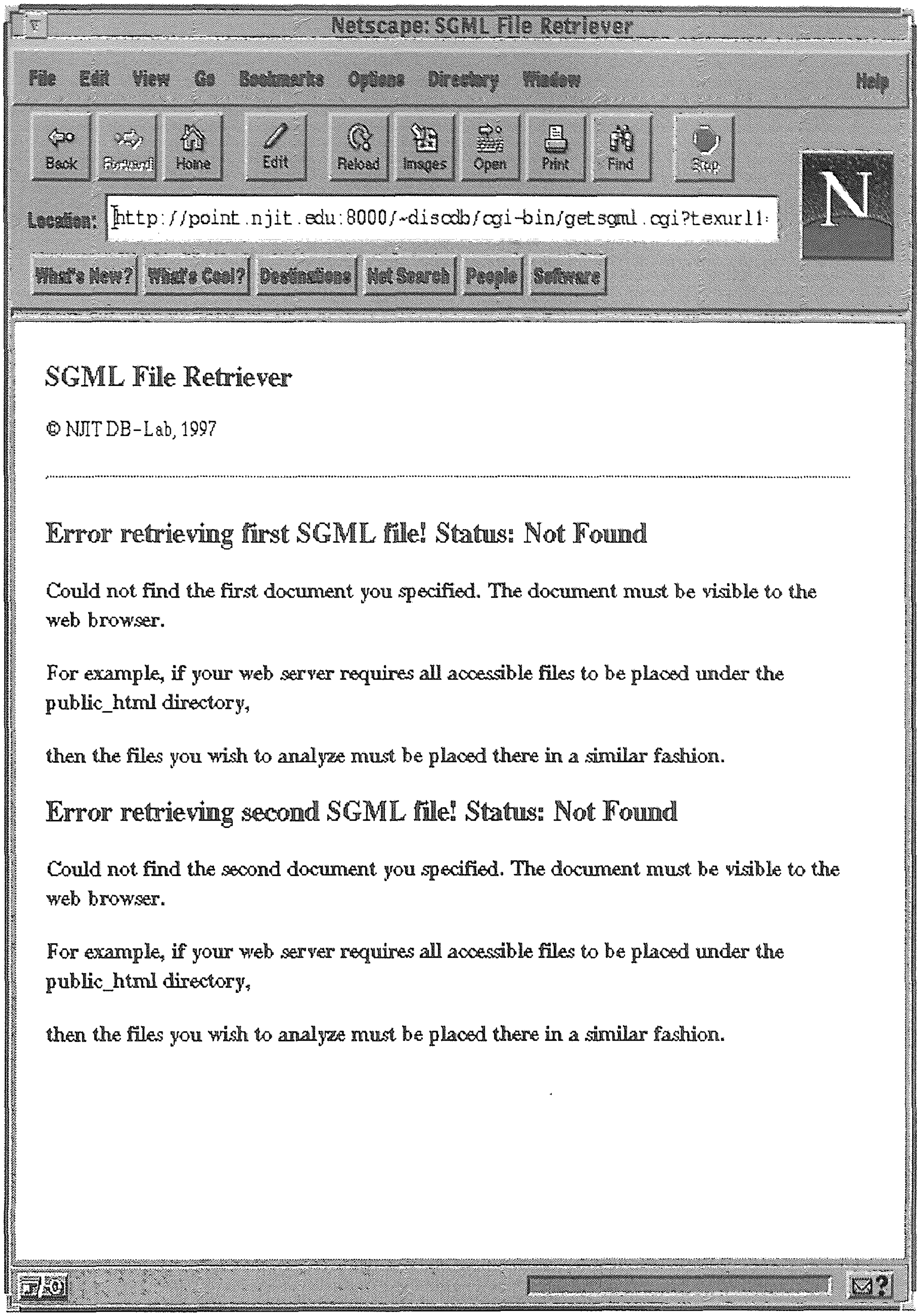
File Comparison System Response

about to read line from stdin

the value of the detail spec is: o
the value of the difference spec is: odiff
all done extracting data
value of diff1 is
value of det1 is
about to call tdsgml via sgml.sh (modified invokeDCT.sh...)

```
This is tdiff for SGML, Version 0.5


-----------------------------------------------------------------------------
Document One: /home/discdb/public_html/cgi-bin/TDSGML_SRC/sgml1.datDocument Two: /home/discdb

Line No    Label                     Line No    Label
-------    -----                     -------    -----
      2                  =         2
      7                     =         7
      7
                  =         7

      8
                  |         8

      9       This memo is be     |         9      This memo is be
     11
                  =        12

     18
                  =        19

     20
                  =        21

     22                       =        23
-----------------------------------------------------------------------------
```

Figure C.3 Output of SGML File Comparison


Netscape: SGML File Retriever
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Edit Reload Images Open Print Find Stop
Location: http://point.njit.edu:8000/~discdb/cgi-bin/getsgml.cgi?texurl1=
What's New? What's Cool? Destinations Net Search People Software

SGML File Retriever

© NJIT DB-Lab, 1997

Error retrieving first SGML file! Status: Not Found

Could not find the first document you specified. The document must be visible to the web browser.

For example, if your web server requires all accessible files to be placed under the public_html directory,

then the files you wish to analyze must be placed there in a similar fashion.

Error retrieving second SGML file! Status: Not Found

Could not find the second document you specified. The document must be visible to the web browser.

For example, if your web server requires all accessible files to be placed under the public_html directory,

then the files you wish to analyze must be placed there in a similar fashion.

Figure C.4 Notification of an SGML File Not Found

C.4 TUTORIAL ON USING TREEDIFF FOR SGML FILE COMPARISON

Following is a tutorial on using TREEDIFF for SGML file comparison on the World Wide Web.

1. Access the TREEDIFF for SGML web page at:

   `http://www.cis.njit.edu/~discdb/csgml.html`

2. Type in the following URLs for the SGML documents:

   `http://point.njit.edu:8000/~discdb/cgi-bin/TDSGML_SRC/memo1.sgm`

   `http://point.njit.edu:8000/~discdb/cgi-bin/TDSGML_SRC/memo2.sgm`

3. When the web page indicating successful retrieval is displayed (see figure C.2, page 63), choose (accept) the default document comparison options and click on 'Begin document comparison'

4. The results will be shown on the next (web) page (see figure C.3, page 64).

REFERENCES

1. G. J. S. Chang, G. Patel, L. Relihan, and J. T. L. Wang, "A Graphical Environment For Change Detection In Structured Documents," in *Proceedings of the 21st Annual International Computer Software and Application Conference*, Washington, DC, August 1997.

2. M. Fuchs, "Semantic Extensions To DSSSL To Handle Trees, or, Why Isn't DSSSL a Tree?," from *http://www.sil.org/sgml/fuchsDSSSL-sgm.txt*, 1996.

3. M. Goossens, F. Mittelbach, and A. Samarin, *The LaTeXCompanion*, Addison-Wesley, Reading, Massachusetts, 1994.

4. J. T. L. Wang, G. -W. Chirn, T. G. Marr, B. A. Shapiro, D. Shasha, and K. Zhang, "Combinatorial Pattern Discovery For Scientific Data: Some Preliminary Results," in *ACM SIGMOD Record*, vol. 23/2, pp. 115–125, 1994.