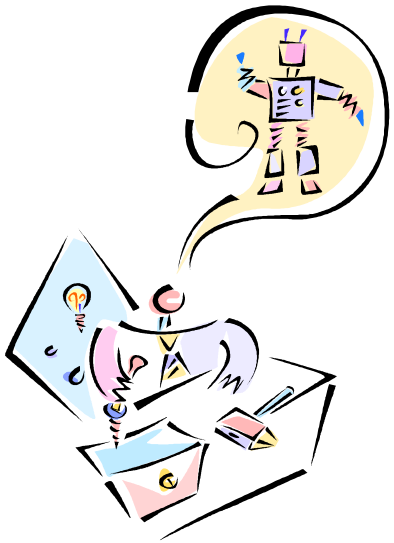
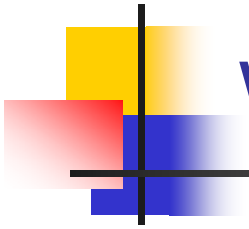
Automated Discovery and Classification of Deep Web Sources



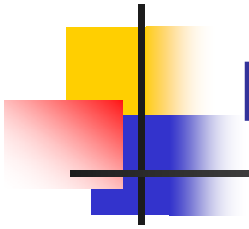
Dr Anne H.H. Ngu
Department of Computer Science
Texas State University-San Marcos



What is a Deep Web Source?

- Surface Web Sources are html pages on the Web that are static and can be indexed and retrieved by traditional search engines.
- Deep Web Sources are dynamically generated html pages from searchable databases. Traditional search engines cannot “see” or retrieve content in deep Web.
- Examples of deep Web sources are BLAST, PubMed, credit checking, and various reservation sites.

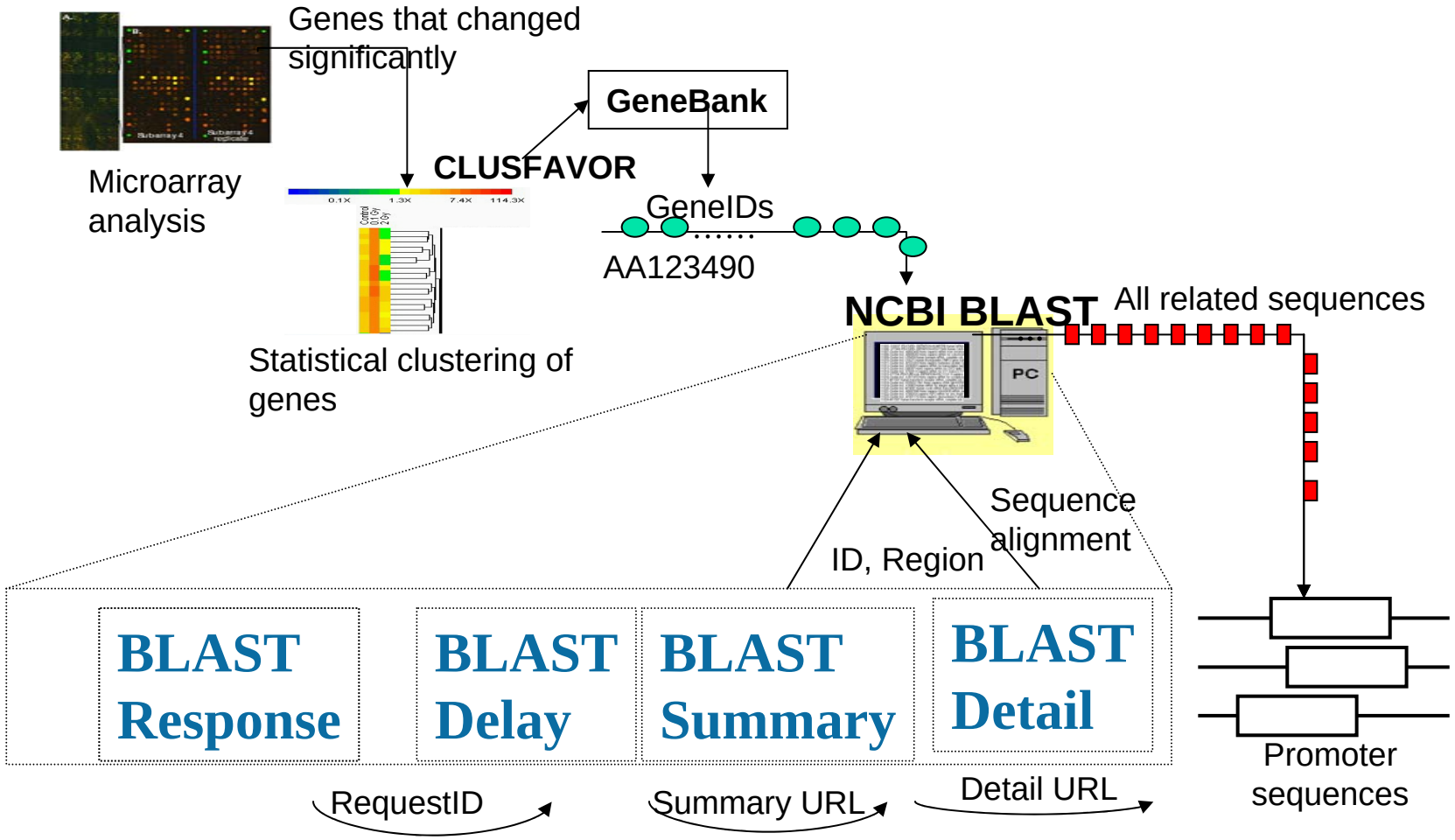
Public information on the deep Web is currently 400 to 550 times Larger than surface Web sources.



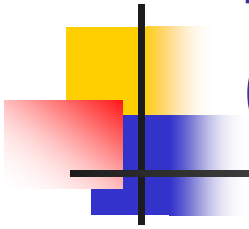
Problems in accessing Deep Web Sources

- The number of available Deep Web sources are increasing at an exponential rate.
- No common interface exist.
- Sources are autonomous and thus can change their interfaces, come and go at will.
- Processes for seeking the desired information in Deep Web source is still very tedious and frustrating (a keyword search using “BLAST sites” return over six millions hit with Google)

Motivating Scenario

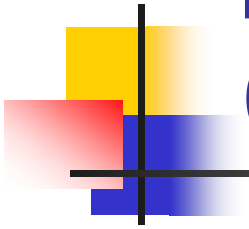


Current approaches to seeking information in Deep Web sources (manual)



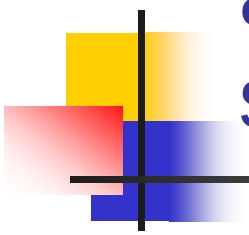
- Must manually identify the set of Deep Web sources to query, then enter the query into each source,
- Must merge the result by hand or create an ad-hoc program to merge them.
- Problems
  - When sources change the way data are presented, new program for merging the results must be recreated.
  - The number of sources used could be incomplete.
  - Important conclusion could be drawn from an incomplete data sets.

Current Approaches to seeking information in Deep Web sources (automated)



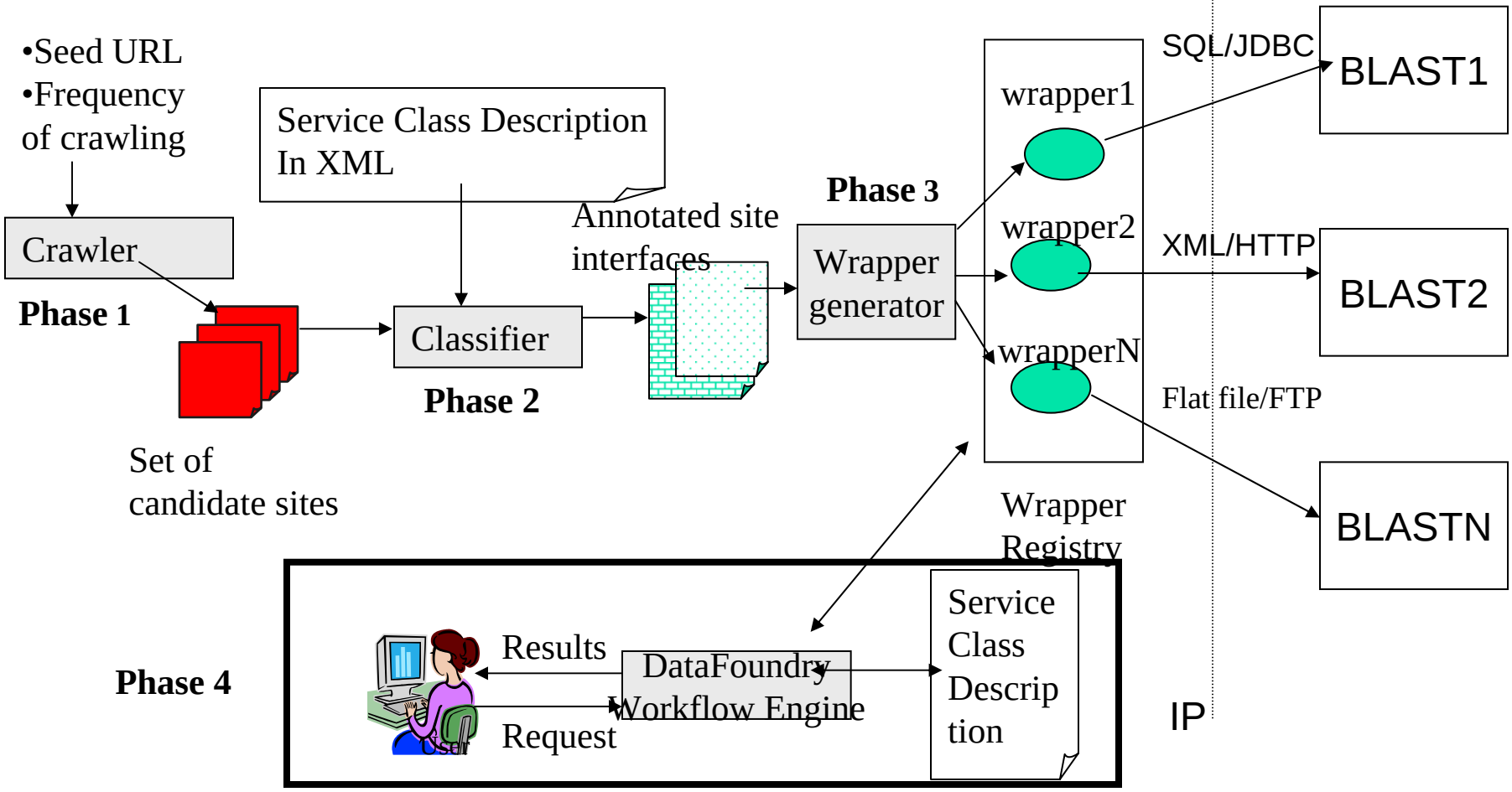
- Forming a federation or creating a virtual view of all the identified sources. This requires a global schema, sophisticated mediators to access the actual source at runtime. New Web source cannot be included automatically.
- Building a data warehouse or creating a materialized view of the data (extract data from all potential sources, store them in a centralized database for querying and analysis)
- Problems
  - Very labor-intensive
  - Not scalable with evolution of Deep Web sources

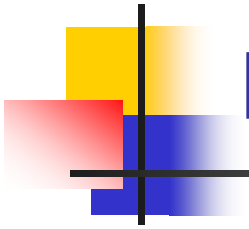
Major challenges in providing a scalable and automated access of Deep Web sources?



- Locating new Deep Web sources
- Evaluating them to determine if they are the Deep Web sources that you are looking for
- Interacting with them to determine how to obtain the desired information
- Constructing wrappers for future automated access to those sources

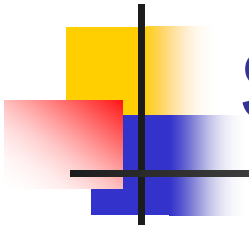
DataFoundry Project at LLNL





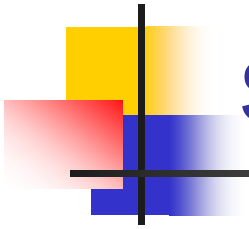
Discovering the deep Web sources

- Require the definition of capabilities of the Web source
- Capabilities can be divided into:
  - Functional component (types of operations performed)
  - Non-functional component (quality of services)
  - Interaction pattern (explicit states that need to be traversed)



Service Class Description

- A *service class* is used to describe the common capability of a class of deep Web sources (BLAST, FlightReservation, BookPurchase) but not necessary a common Web interface
- A *service class* is specified using combination of XML Schema and regular expression



Service class main types of information

- Data types – the types expected in input parameters, results (output) as well as any unique intermediate types that may appear in a class of Web sources.
- Input parameters – common names used in html form for posing a request to the Web source.
- Example data – typical instances of data used to evaluate the source.
- Control Flow – the generic navigation pattern of a class of Web source for a particular operation.

Example data types definition in BLAST Service Class

```
<type name="DNASequence" type="string" pattern="[GCATgcat-]+" />
<type name="AlignmentSequenceFragment" >
    <element name="AlignmentName" type="string" pattern=".
{1,100}:" />
    <element type="whitespace" />
    <element name="m" type="integer" />
    <element type="whitespace" />
    <element name="Sequence" type="DNASequence" />
    <element type="whitespace" />
    <element name="n" type="integer" /></type>
<type name="AlignmentString">
    <element DNAAlignmentString" type="string" pattern="\s+\|+[| +]
*" /></type>
<type name="Alignments"
      <element name="QueryString" type="AlignmentSequenceFragment"
      <element type="string" pattern="\s*" />
      <element type="AlignmentString" required="true"/>
      <element type="string" pattern="\s*" />
      <element name="SequenceString"
type="AlignmentSequenceFragment" /></type>
<type name="SummaryResults">
    <choice>
          <element type="Alignments" />
          <element type ="EmptyDNABLAST"/>
    </choice>
</type>
```

SummaryResults

```
Query:   179 GGCTTCTACACCAAAGTGCTCAACTACGTGGACTGGAT 142
             ||  | |||||||| ||| || | ||||||||||||||
Sbjct:     3 GGTGTTTACACCAACGTGGTCGAGTACGTGGACTGGAT 40
```

Example of an Alignments

*** No Hits ****

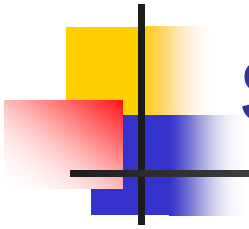
Example of an EmptyDNABLAST

Example control flow definitions in BLAST Service Class

```
<controlgraph name="BLASTN">
    <vertices>
      <vertex name="start" type="HTMLform"/>
      <vertex name="end" type ="SummaryResults" />
    </vertices>
    <edges>
      <edge origin="start" destination="end" />
    </edges>
</controlgraph>
```

Example Sample Data Definitions in BLAST Service Class

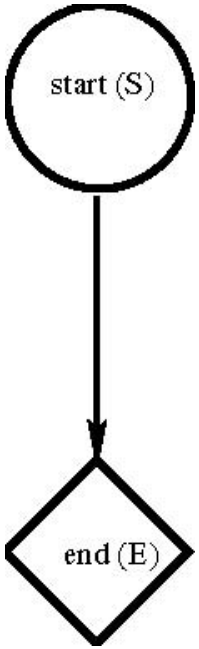
```
<example>
 <arguments>
  <argument required="true">
   <name>sequence</name>
   <type>DNASequence</type>
   <hints>
    <hint>sequence</hint>
    <hint>query</hint>
    <hint>query_data</hint>
    <inputType>text</inputType>
   </hints>
   <value>TTGCCTCACATTGTCACTGCAAAT
        CGACACCTATTAATGGGTCTCACC
   </value>
  </argument>
  <argument required="false">
   <name>BlastProgram</name>
   <type>string</type>
   <hints>
    <hint>program</hint>
   </hints>
   <value>blastn</value>
  </argument>
 </arguments>
</example>
```



Source capability classification process

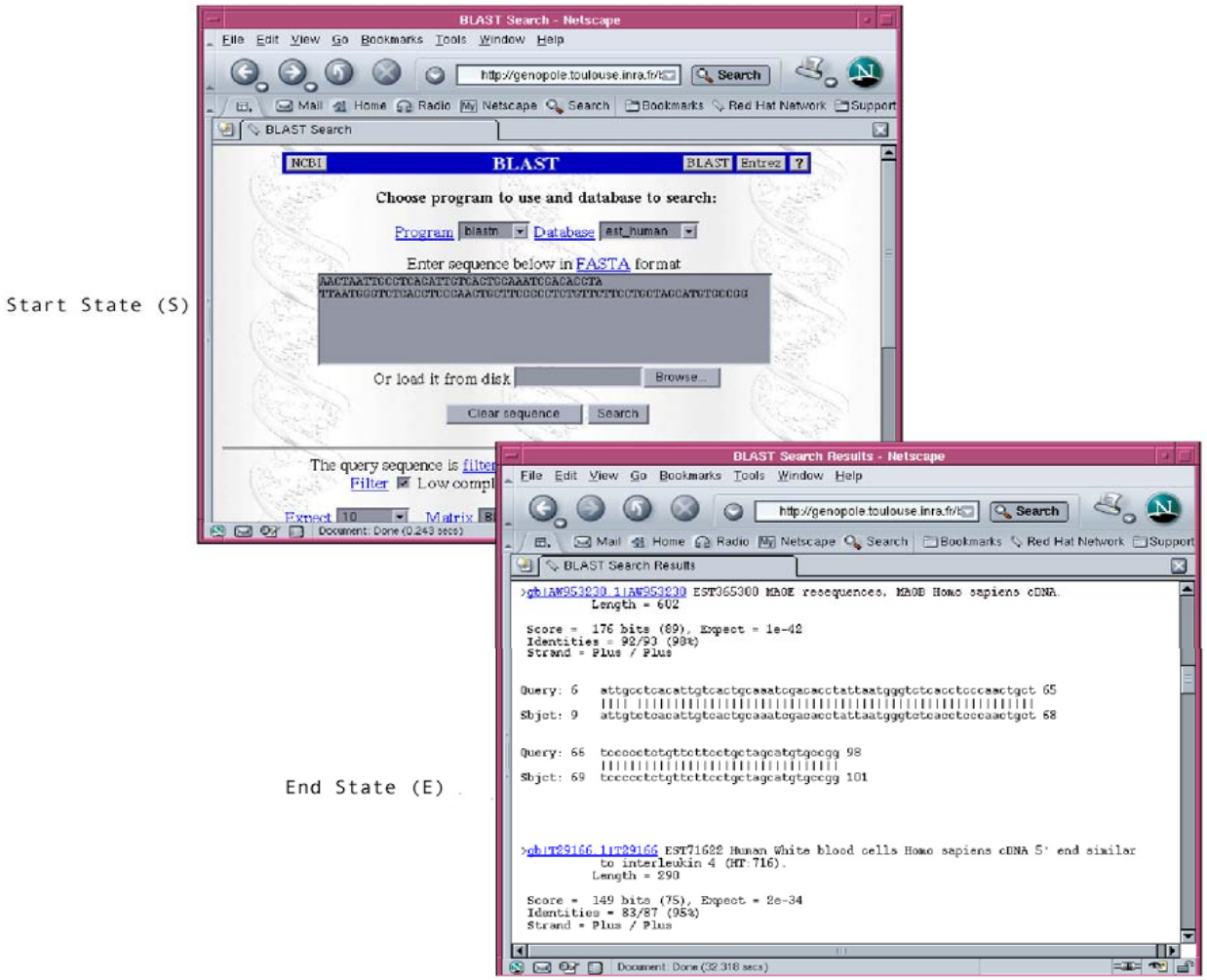
- Locating Web Interface – done by a crawler agent which locates an HTML form with a text input parameter, labelled with ‘sequence’, ‘query_data’, or ‘query’ in its name.
- Evaluating the source – check whether the interface conforms to the capability specified in the Service Class Description (SCD).

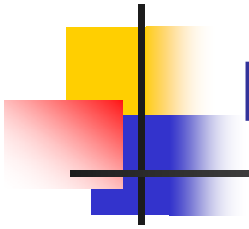
Evaluation of the source



- Guided by the control flow graph in SCD.
- First match the start page of the source against the start state in the control flow graph.
- If there is a match, then generates a series of test queries to probe the sites based on the sample data specified in *SCD*.
- The evaluation process continues either the site matches one of the end states in the control flow graph or there are no more possible queries to try.
- If a site is positively identified, the steps used to reach the end state of the control graph along with the input parameters used to probe the site will be saved.

Example of a potential deep Web source that match BLAST Service Control Flow Graph

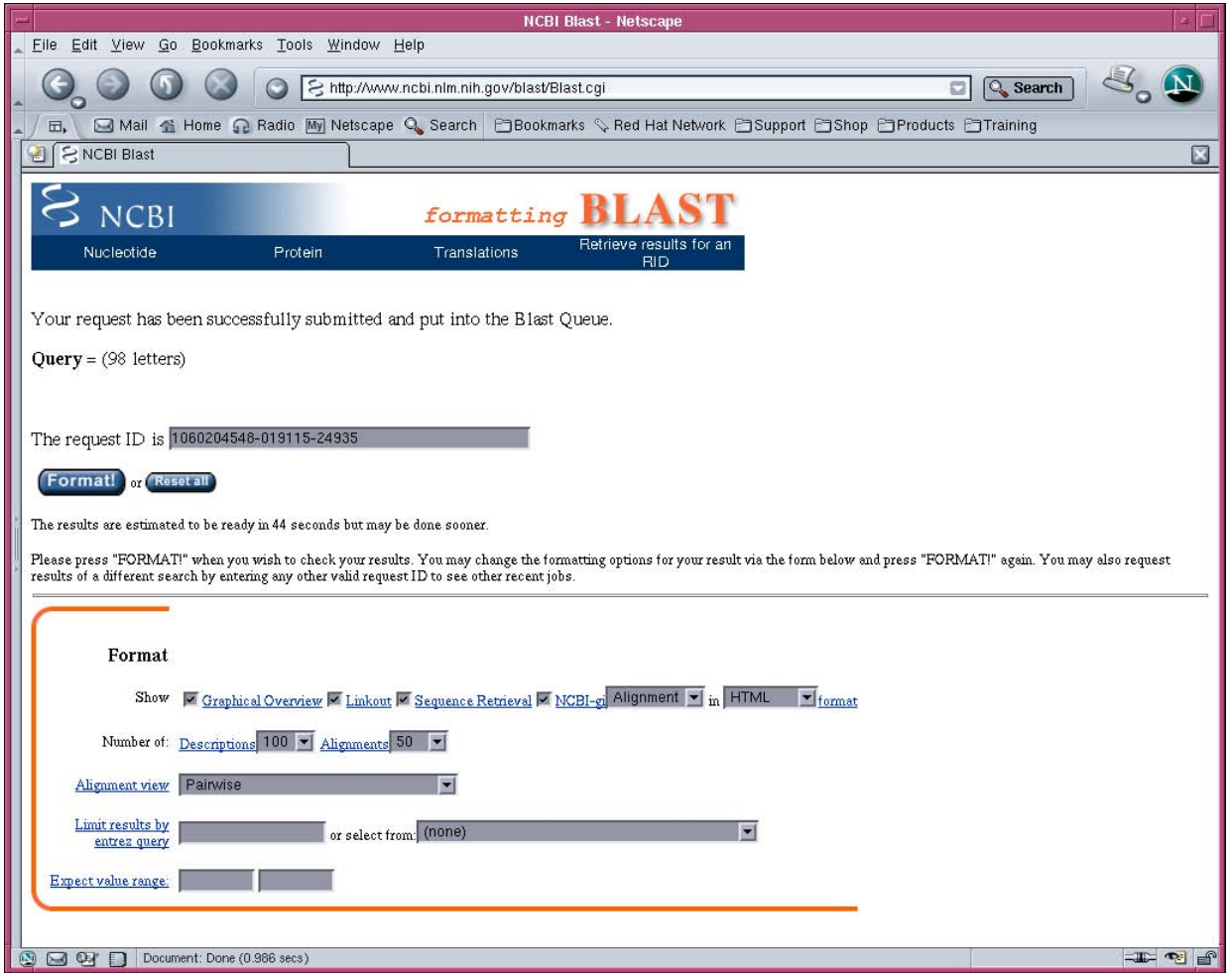




Discovery of BLAST sites using SCD

- Able to correctly classify two third of 150 BLAST sites using SCD of 150 lines.
- Failed to classify important sites such as NCBI BLAST because its control flow is more than the simple start and end states. It utilizes intermediate page that need to be traversed to reach the *end* state.
- Sites may have its own specific interaction pattern (intermediate pages) due to site specific implementation. These should not be specified in the SCD.
- We use the term indirection page for site specific interaction pattern
- The indirection pages need to be inferred during classification process.

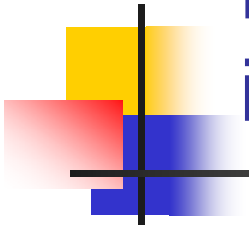
Example of an Indirection Page



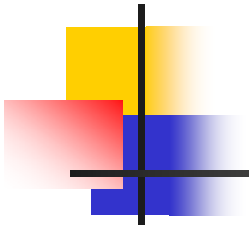
Different types of Indirection Page

Site No	Site URL	Type of Indirect Page
1	http://www.genedb.org/genedb/dicty/blast.jsp	Retrieve button and links
2	http://www.sgn.cornell.edu/cgi-bin/SGN/blast/blast search.pl	Click to view result link
3	http://pbil.iniv-lyon1.fr/BLAST/blast nuc.html	Click here to see your results → Click reload button to check status
4	http://zeon.well.ox.ac.uk/git-bin/blast2	forms &links → Refresh → Click here → output page
5	http://www.rtc.riken.go.jp/jouhou/HOMOLOGY/blast	Check your entry → new pop-up window for email result
6	http://www.ncbi.nlm.nih.gov/blast/Blast	Format button & links
7	http://www.bioinfo.org.cn/lmh/blastlmh.html	Press it button
8	http://www.sanger.ac.uk/HGP/blast server.shtml	Retrieve result button & links
9	http://genoplante-info.infobiogen.fr/pise/blast2 gpi.html	Multiple forms & links → email
10	http://www.ebi.ac.uk/blast2	Refresh → forms, links & result summary page

Naïve approach to indirection page identification



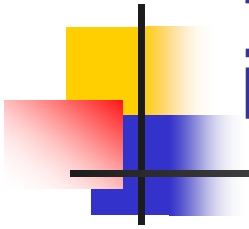
- check all the links exhaustively
  - Computationally very expensive
  - Treating all out bound links as equal
  - Flooding a site with too many http requests



Indirection Page detection

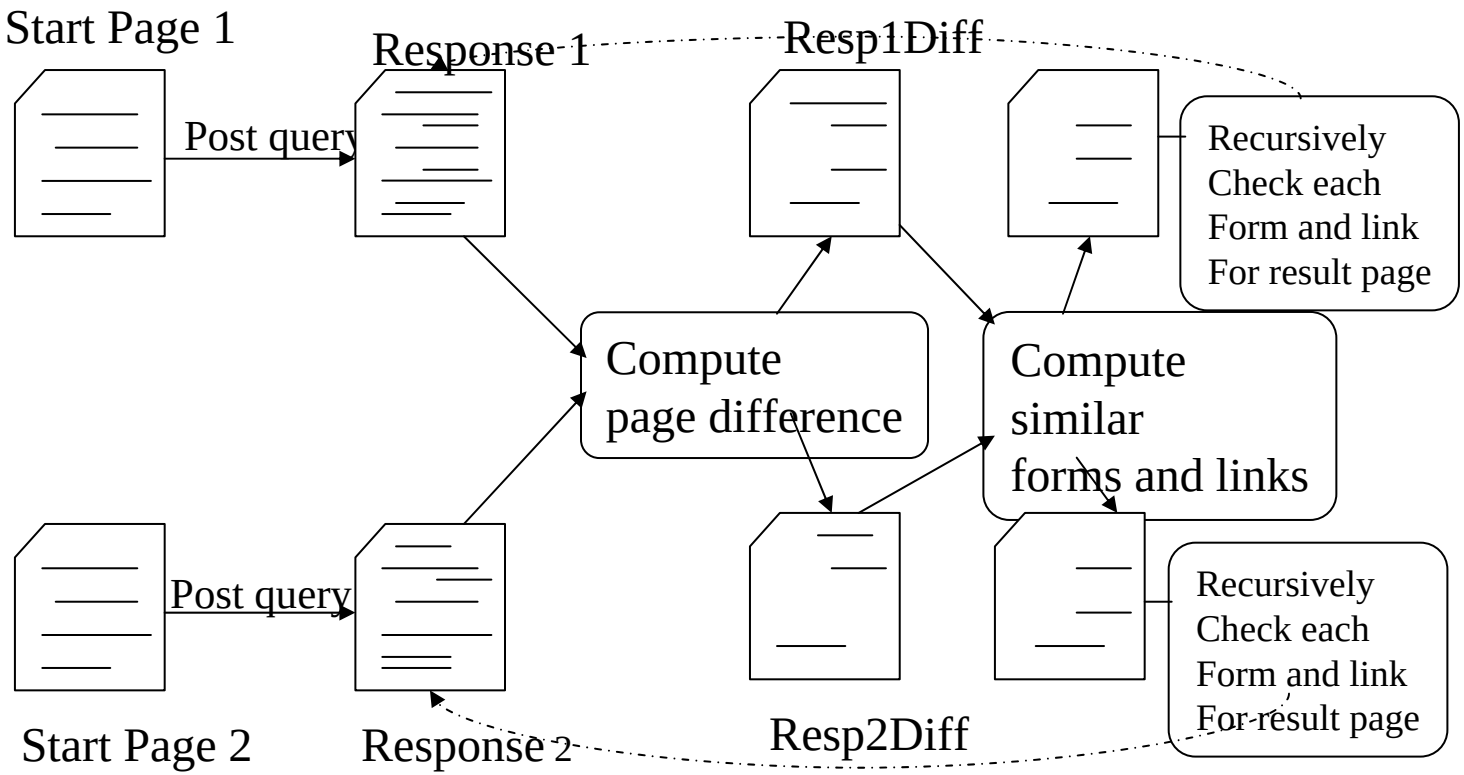
- Require a scheme to prioritize the outbound links
- Do not want to get into expensive natural language understanding to figure out links of indirection pages
- The technique must be robust against various type of indirection pages that can be generated by a web source

PageDiff approach for discovery of indirection pages



- Only outbound links that are dynamically generated as a response to a query are potential indirection pages.
- If we post two similar requests to the same Web interface and compute the difference in the outbound links between the two response pages, the resulting set (typically very small) will contain the outbound links that are dynamically generated.
- It is possible to quickly follow all the outbound links in this set to identify the indirection page.

PageDiff Algorithm



Examples of similar links

http://www.rcsb.org/pdb/cgi/explore.cgi?pid=940410589088 65&page=0&pdbId=1B20
http://www.rcsb.org/pdb/cgi/explore.cgi?pid=960110589090 35&page=0&pdbId=1AOO.

```
<a href="http://www.xxx.yyy.org/blast/A123456.html">blast</a>
<a href="http://www.xxx.yyy.org/blast/A125678.html">blast</a>
```

```
<a href="http://www.xxx.yyy.org/blast/tmp/A123456/index.html">blast</a>
<a href="http://www.xxx.yyy.org/blast/tmp/A234567/index.html">blast</a>
```

Experimental Evaluation

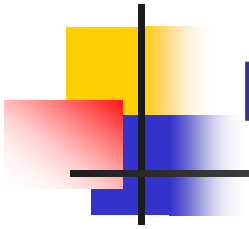
- The prototype is implemented in Java.
- Interaction with deep Web source is handled by HttpUnit user agent library.
- Precision and Recall in IR is used to measure the performance of our prototype.
- Two set of experiments (BLAST and Bio-KeyWord) are performed.

$$\text{Recall} = \frac{relevantSitesIdentified}{totalrelevantSites}$$

$$\text{Precision} = \frac{relevantSitesIdentified}{totalSitesIdentified}$$

BLAST Experimental Result

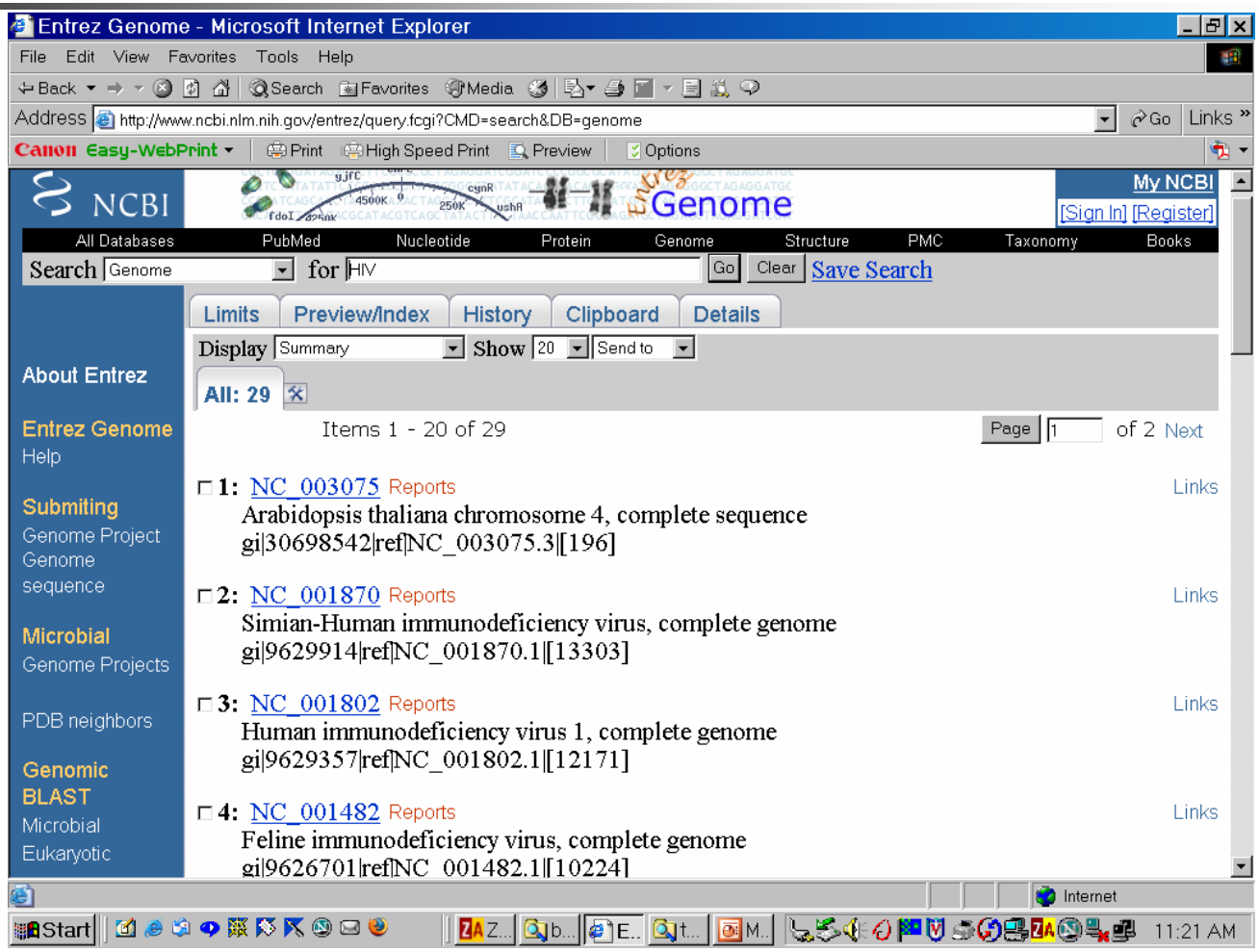
Data Set	True Negative	True Positive	True Positive with Indirection	False Negative	False Positive	Recall	Precision
Test Set	8	18	3	6	0	77.7 %	100%
Experimental Set	46	64	5	23	0	75.0 %	100%



Bio-KeyWord Sources

- Bio-keyWord sources are keyword-based bioinformatics search sites for protein and nucleotide sequences.
- A deep Web source is considered to be Bio-KeyWord site if it allows user to input a keyword (HIV, Cancer) and returns the result in an HTML page with a list of pointers to files that have detailed information for the protein or nucleotide sequences relevant to that keyword.
- Each pointer in the returned HTML page is an HTML link indexed by either protien ID code (protein sequence) or accession number (nucleotide).
- Bio-KeyWord sites are completely different in input requirement and the result summary page as compared to BLAST.
- A new BioKey SCD with 120 lines of code is used.
- The same mechanism can be used to discover and classify Bio-KeyWord sources.

Sample output page from a Bio-keyWord Source

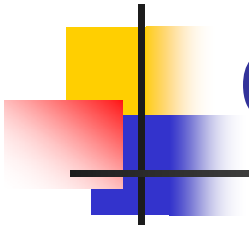


BioKeyWord SCD

```
<type name="AccessionNumberType“
       type="string“ pattern="([A-Z]{2}_?\d{6}|[A-Z]\d{5})"/>
<type name="AccessionNumber">
      <elementtype="string“ pattern="[^A-Za-z0-9]*?"/>
      <elementtype="AccessionNumberType"/>
      <elementype="string“  pattern="[^A-Za-z0-9]"/>
</type>
<type name="HREF">
    <element
type="string“pattern="\x3C(a|A)[^\x3E]*(href|HREF)=[\x22\x27]"/>
    <element name="Link“ type="string“ pattern="[^\x22\x27]+"/>
    <element type="string“ pattern="[\x22\x27][^\x3E]*\x3E"/>
</type>
<typename="NucleotideLink">
    <element type="HREF"/>
    <element name="accessionNo“ type="AccessionNumber"/>
</type>
```

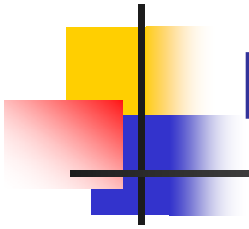
Bio-KeyWord Experimental Results

Data Set	True Negative	True Positive	True Positive with Indirection	False Negative	False Positive	Recall	Precision
Test Set	10	7	1	1	1	87.7 %	87.7%
Exp. Set 1	81	3	2	2	1	71.4 %	83.3%
Exp. Set 2	82	5	0	0	0	100%	100%
Exp. Set 3	43	2	0	1	0	63.3 %	100%



Conclusions

- BLAST sources and Bio-KeyWord sources have completely different Web interfaces and completely different input and output requirement, the same mechanism can be used to identify them based on their respective SCD.
- DNA alignments are very unique to BLAST sources and thus we can achieve 100% precision.
- The protein code is unique within the respective domain, but an arbitrary Web source can also generate a four letter acronym used for protein code identification, thus we achieve lower precision with BioKeyWord.
- SCD is a powerful paradigm for classifying Web sources whose output exhibit a regular pattern that can be described using a regular expression.
- SCD is not effective for classifying Web sources whose output pattern is irregular and can all be literally defined using the same regular expression. For example, a publication web source.



Future Works

- Automate the creation of SCD using supervised machine learning technique. A paper called “Automated Generation of Data Types for Classification of Deep Web Sources” has been submitted to Data Integration in the Life Sciences Workshop.
- Extend the detection of indirection page to those that also requires specialized user interaction.
- Extend SCD definition to include the quality of service.