26. Structure-Based Models [2]

INFO 202 - 26 November 2008

Bob Glushko

Plan for Today's Class

IR and the Document Type Spectrum

XPath and XML Queries

Benefits and Challenges for Structural Models

Search Engine Optimization

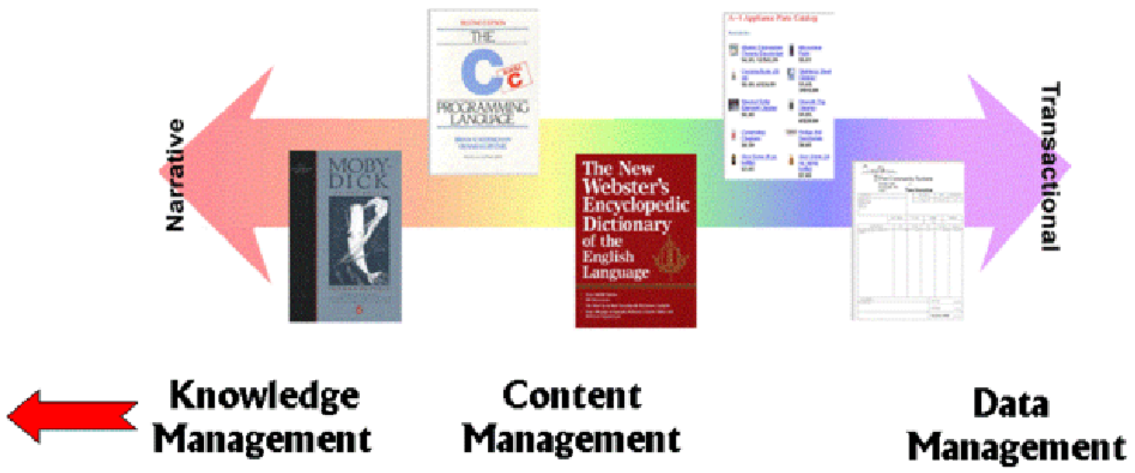
Why We Want Structure Based Retrieval

Documents aren't just bags of words; they can have a great deal of internal structure and content encoding

But most IR models don't use anything other than document-level statistics about term occurrence

The use of XML for encoding document models and instances shows where structure can be used to great advantage in IR to add value beyond text retrieval

The Document Type Spectrum


Narrative
Transactional
Knowledge Management
Content Management
Data Management
MOBY DICK
THE C PROGRAMMING LANGUAGE
The New Webster's Encyclopedic Dictionary of the English Language

Document Types in the Spectrum

"Narrative" Document Types

- Novels, Reports, Brochures, Academic Monographs, Textbooks, ...

"Hybrid" Document Types

- Catalogs, Dictionaries, Encyclopedias, Technical Manuals, ...

"Transactional" Document Types

- Orders, Invoices, Payments, Transcripts, Bill of Materials, ...

Structure in Narrative Document Types

Narrative document types have relatively little internal structure

- They may have some Dublin-Core-like metadata (title, author, date, etc.)
- They may have some presentational structures (chapter, section, etc)
- But they rarely have internal structure that makes content-type distinctions

So while it is possible to label sections or chapters of the text with titles or assign index terms to them, it is seldom useful to treat those parts as specialized types of content

Put another way, the content of narrative documents is weakly datatyped – "just text" (PCDATA in XML content models)

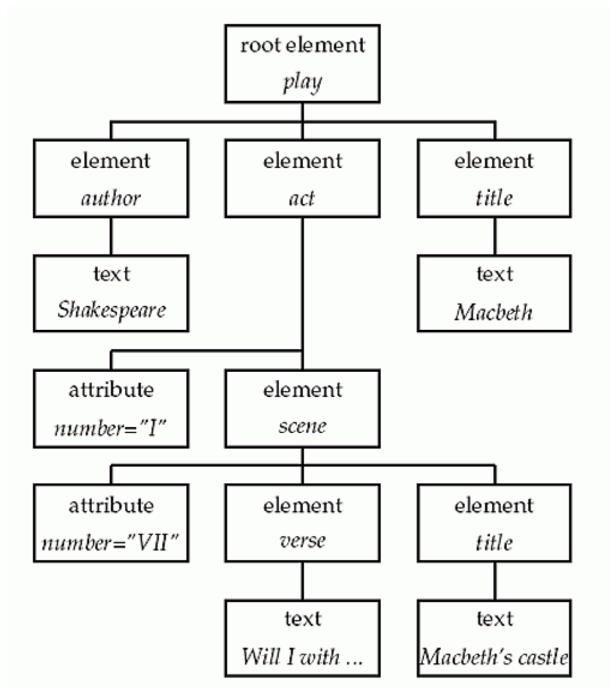
An Exception That Proves the Rule

There are some notable exceptions for specialized narrative document types like "theatrical play" where there are highly conventional structural divisions and presentation rules that can increase precision in IR

Or maybe it is more correct to say that the specialized use cases or context of use for plays requires us to impose more structure than most narrative document types need

```
<play>
<author>Shakespeare</author>
<title>Macbeth</title>
  <act number="I">
    <scene number="VII">
    <title>Macbeth's castle</title>
       <verse>Will I with wine and wassail...</verse>
   </scene>
 </act>
</play>
```

Hierarchical Structure



Structure in Hybrid Document Types

Hybrid document types are often called "semi-structured" because they have more structure than narrative ones but less than the fully-structured transactional ones

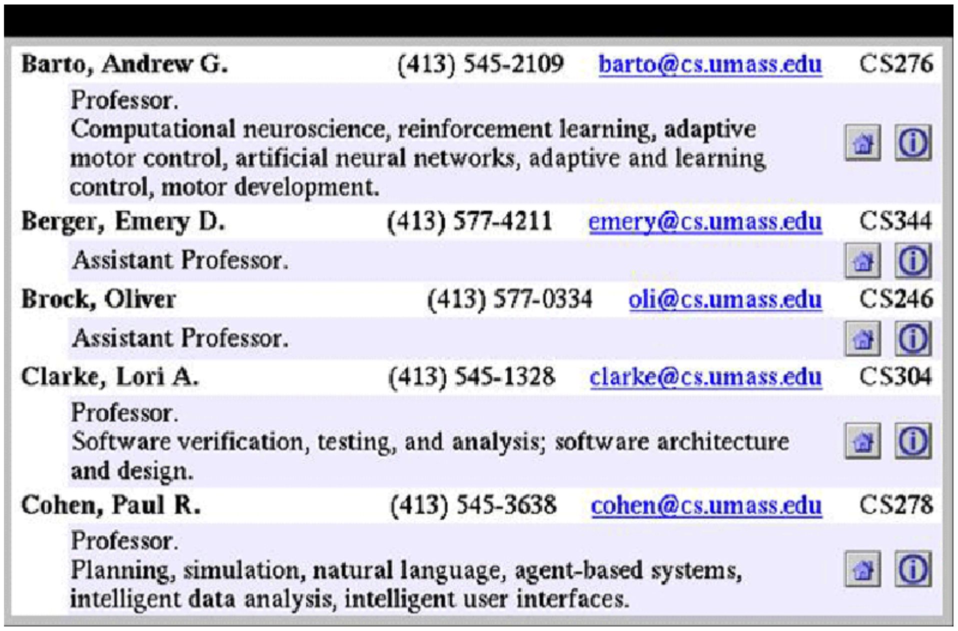
Hybrid document types exhibit more regularity in data content and structure than pure narrative types, but some presentation structure remains important because these documents are used by people

Key design question is how many of the content types are explicitly marked or tagged in the document

The extent of explicit structure in semi-structured document types embodies the IO vs IR tradeoff we've discussed all semester

In XML modeling terms, this is how much "mixed content" remains -- blocks or paragraphs of text that isn't differentiated by content type

A Hybrid Document Instance: Directory



Barto, Andrew G.	(413) 545-2109	barto@cs.umass.edu	CS276
Professor. Computational neuroscience, reinforcement learning, adaptive motor control, artificial neural networks, adaptive and learning control, motor development.			
Berger, Emery D.	(413) 577-4211	emery@cs.umass.edu	CS344
Assistant Professor.			
Brock, Oliver	(413) 577-0334	oli@cs.umass.edu	CS246
Assistant Professor.			
Clarke, Lori A.	(413) 545-1328	clarke@cs.umass.edu	CS304
Professor. Software verification, testing, and analysis; software architecture and design.			
Cohen, Paul R.	(413) 545-3638	cohen@cs.umass.edu	CS278
Professor. Planning, simulation, natural language, agent-based systems, intelligent data analysis, intelligent user interfaces.			

"Directory" Document Type with "Mixed Content"

```
<Professor>
  <Name>Barto, Andrew G.</Name>
  <Phone>(413) 545-2109</Phone>
  <E-mail>barto@cd.umass.edu</E-mail>
  <Office>CS276</Office>
  <MoreInfo>
   Professor. Computational neuroscience, reinforcement learning, ...
  </MoreInfo>
</Professor>
```

More Structured "Directory" Document Type With No Mixed Content

```
<Professor>
  <Name>
    <FirstName>Andrew</FirstName>
    <MiddleInitial>G</MiddleInitial>
    <LastName>Barto</LastName>
  </Name>
  <Phone>
    <AreaCode>413</AreaCode>
    <Number>543-2109</Number>
  </Phone>
  <E-mail>barto@cd.umass.edu</E-mail>
  <Office><Building>CS</Building><Room>276</Room></Office>
  <MoreInfo>
    <Rank>Professor</Rank>
    <Interests>
       <Interest>computational neuroscience</Interest>
       <Interest>reinforcement learning</Interest>
        ...
   </Interests>
  </MoreInfo>
</Professor>
```

SylViA's Choice

```
<reading>
    Susan Dumais. Data-driven approaches to information access.
    Cognitive Science, 27(3), 491-524, 2003.
    http://www.sims.berkeley.edu/courses/is202/f06/Readings/...
</reading>


<journalReading>
  <articleTitle>Data-driven approaches to information access</articleTitle>
  <articleAuthor>
    <givenName>Susan</givenName>
    <familyName>Dumais</familyName>
   </articleAuthor>
  <journalTitle>Cognitive Science</journalTitle>
  <volume>27(3)</volume>
  <publishDate>2003</publishDate>
  <pageSpan>491-524</pageSpan>
  <url>http://www.sims.berkeley.edu/courses/is202/f06/Readings/...</url>
</journalReading>
```

"Transactional" Document Types

Transactional documents are completely and regularly structured, with prescriptive document type models in which every piece of information is "strongly typed"

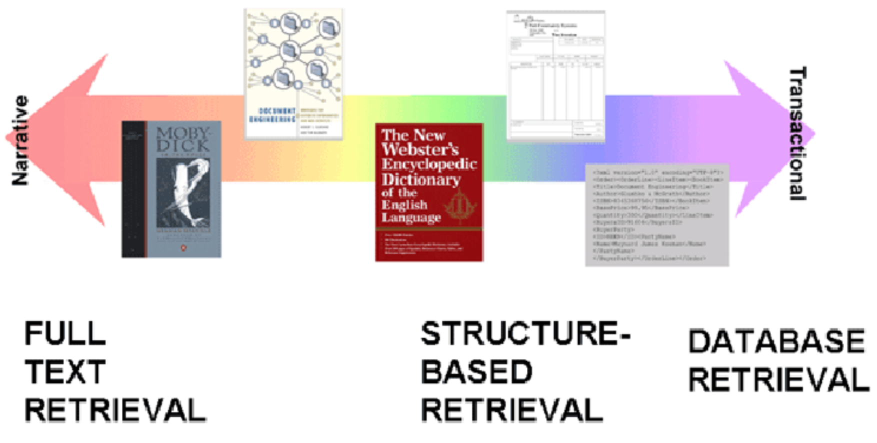
The instances are typically created by automated processes or captured in highly-structured forms

Because every piece of information is explicitly marked up or delimited, it is separately retrievable

A Transactional Document Instance

```
<DespatchLine>
<ID>1</ID>
  <DeliveredQuantity quantityUnitCode="PKG">5</DeliveredQuantity>
  <OrderLineReference>
    <BuyersLineID>1</BuyersLineID>
  </OrderLineReference>
  <Item>
    <Description>Book "Document Engineering"</Description>
    <SellersItemIdentification>
      <ID>32145-12</ID>
    </SellersItemIdentification>
    <BasePrice>
      <PriceAmount amountCurrencyCodeListVersionID="0.3"
       amountCurrencyID="USD">32.50</PriceAmount>
    <BasePrice>
  </Item>
</DespatchLine>
```

IR Models and the Document Type Spectrum



Storing and Retrieving XML

Highly-structured transactional information encoded in XML can be stored in a relational database, but XML poses other problems

The content of databases is often exported or retrieved in XML for publishing information or passing it to applications

And of course it is easy to store XML documents in a database as "text" but then the "XML-ness" of the content is of no use in IR

So by "XML Retrieval" we generally are focusing on hybrid or semi-structured document types that aren't the "bread and butter" of relational databases

Semi-structured document types NOT encoded in XML aren't the focus of much attention in IR

XPath

A standard way of addressing parts of XML documents

Defines the structures and patterns used by XML transformations, queries, and forms

Similar in concept to addressing files on the filesystem, i.e. at a UNIX shell or MS-DOS command prompt

Key idea is to view an XML document as a tree of information items called "nodes" - this is more abstract than thinking of it as a stream of marked-up text

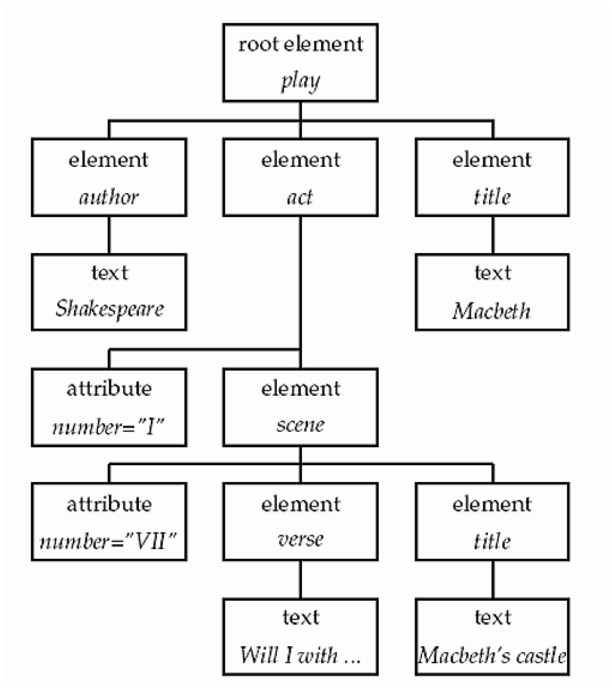
The Node Tree

XPath describes the locations of addresses of parts of XML documents by navigating through the "node tree" along a "node axis"

There are seven types of nodes, corresponding to the different kinds of "stuff" in XML documents (most important are "element," "attribute," and "text")

There are thirteen different axes that specify different ways of following relationships among the nodes (the default is "child" -- meaning, look down the tree at the nodes directly linked as children)

The Tree



The Node Axes [1]

There are thirteen different *axes* that define different directions of "walking the node tree" depth-first starting from the *context node;*

Depth-first means visiting all the children recursively throughout the entire document, shown using the numbering of the nodes in the following graphs

The Self Axis identifies the context node

The Child
Axis identifies the children of the context node. This is the default so if the axis is omitted the child axis is assumed

The Attribute Axis identifies the attributes of the context node

The Parent Axis identifies the parent of the context node

The Node Axes [2]

The Following
Axis identifies all nodes after the context node in document order, excluding its attributes and descendants

The Following-Sibling
Axis identifies all nodes that follow the context node in document order and that have the same parent

The Preceding
Axis identifies all nodes before the context node in document order, excluding its ancestors and any attributes

The Preceding-Sibling
Axis identifies all nodes before the context node in document order and that have the same parent

Finding Documents With Structured Queries (Bourret 4.2.2)

Find all books that Maria Lopez wrote:

```
for $b in collection("books")
where $b//Author="Maria Lopez"
return $b
```

Find all articles written after June 1, 2004 with the words "presidential election" in the title:

```
for $a in collection("articles")
where $a//Date > 2004-06-01 and
      fn:contains($a//Title, "presidential election")
return $a
```

Find all procedures with more than seven steps:

```
for $p in collection("procedures")
let $s := $p//Step
where fn:count($s) > 7
return $p
```

Retrieving Information With Structured Queries (Bourret 4.2.2)

From a procedure for synthesizing a compound, list the required chemicals:

```
for $p in collection("procedures")
return
   <Chemicals procedure="{$p/Title}">
      {$p//Chemical}
   </Chemicals>
```

Find maintenance procedures for a specific spare part of a specific airplane with a specific effectivity:

```
let $today = fn:current-date()
for $proc in collection("maintenance_docs")//Procedure
let $p = $proc//Part
where $p = "AX723" and $p//AppliesTo = "Model 1023i"
      and fn:date-greater-than($today, $p//EffectivityStart)
      and fn:date-less-than($today, $p//EffectivityEnd)
return $proc
```

The Structured Document Retrieval Principle

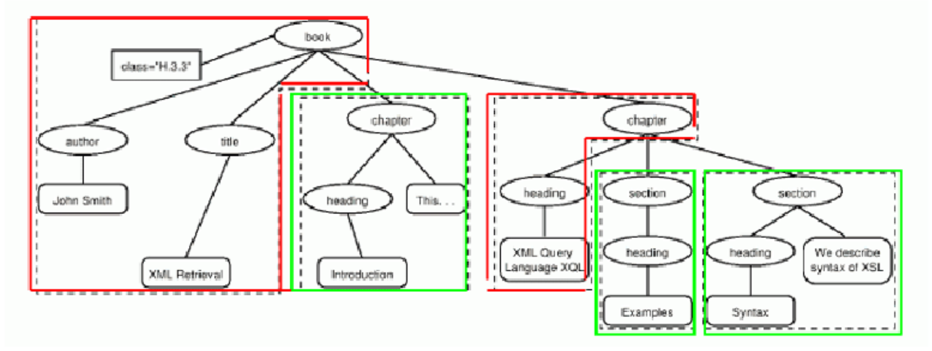
But if documents have useful internal structure, you should retrieve the most specific part of a document you can. So a query for *Macbeth* might retrieve everthing after the top <title> element, but a a query for *Macbeth's castle* might retrieve the document starting with the <title> element that is a child of <scene number="VII">

```
<play>
<author>Shakespeare</author>
<title>Macbeth</title>
  <act number="I">
    <scene number="VII">
    <title>Macbeth's castle</title>
       <verse>Will I with wine and wassail...</verse>
   </scene>
 </act>
</play>
```

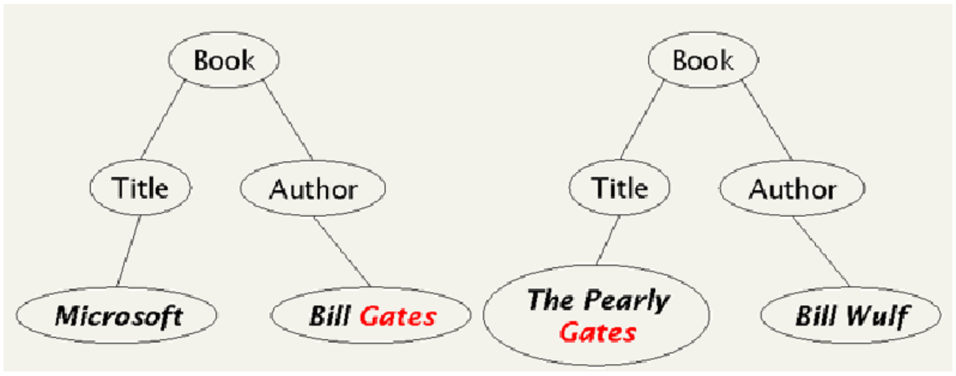
What's The Indexing Unit for Structured Documents?

If we index all the components that would make sense to return as a match to a query, we end up with overlapping units

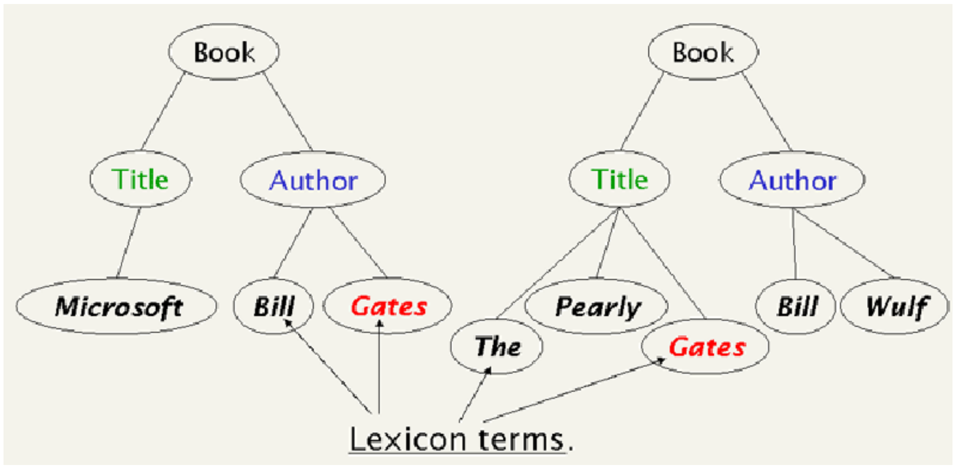
But if we group nodes in the document tree into non-overlapping "pseudo-documents" what gets retrieved may not make intuitive sense



Distinguish These Two Cases



The XML Content Representation



Encoding the Two "Gates" in a Vector Model

Vector models are the tried and tested framework for term-based retrieval

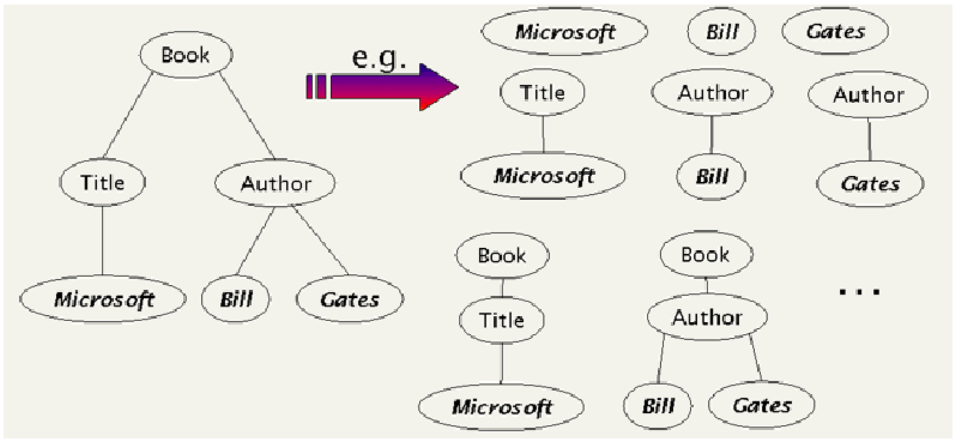
But if we use a single axis for "Gates" we can't do structure-based retrieval

We must separate out the two occurrences, under Author and Title

So axes must represent not only terms, but something about their position in an XML tree

Subtrees and Structure

But how many subtrees of the document contain at least one lexicon term?



Structural Terms

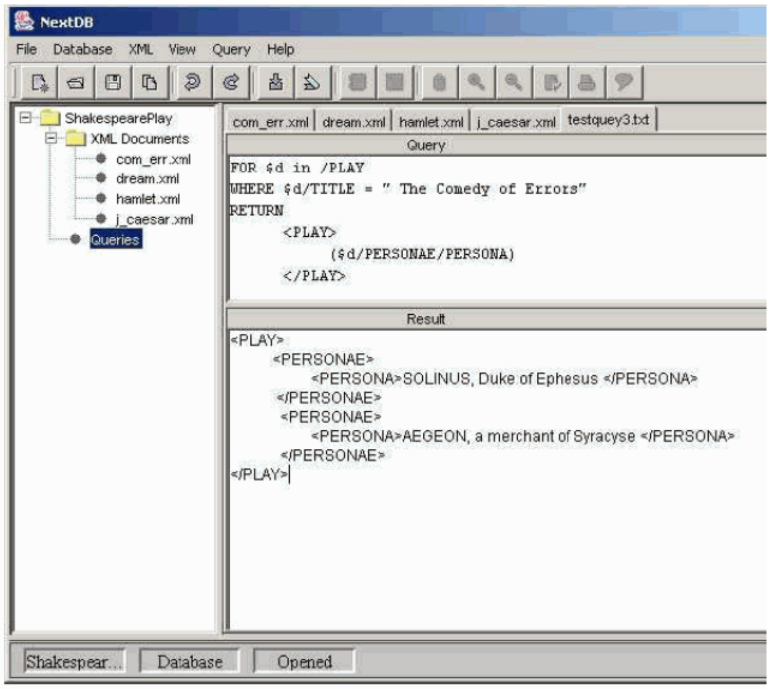
In the "Gates" example there are more than 8 subtrees

We create one axis in the vector space for each distinct structural term

We'll compute weights based on frequencies for number of occurrences of each structural term (just as we had tf)

This is getting very complicated...and to process queries, we need to factor them into structural terms

User Interfaces for Structural Search


NextDB
File Database XML View Query Help
ShakespearePlay
XML Documents
com_err.xml
dream.xml
hamlet.xml
j_caesar.xml
Queries
com_err.xml dream.xml hamlet.xml j_caesar.xml testquey3.txt
Query
Result
Shakespear... Database Opened

```
FOR $d in /PLAY
WHERE $d/TITLE = " The Comedy of Errors"
RETURN
      <PLAY>
            ($d/PERSONAE/PERSONA)
      </PLAY>
```

```
<PLAY>
     <PERSONAE>
          <PERSONA>SOLINUS, Duke of Ephesus </PERSONA>
     </PERSONAE>
      <PERSONAE>
          <PERSONA>AEGEON, a merchant of Syracyse </PERSONA>
      </PERSONAE>
</PLAY>
```

Models of Information Retrieval -- Concluding Thoughts

The core problems of information retrieval are finding relevant documents and ordering the found documents according to relevance

The IR model explains how these problems are solved:

- ...By specifying the representations of queries and documents in the collection being searched
- ...And the information used, and the calculations performed, that order the retrieved documents by relevance

Different IR models solve these problems in different ways; the better they solve it, the more computationally complex they are, so there are tradeoffs.

The Motivation for "Search Engine Optimization"

Transaction log studies show that search engine users tend to look only at the first page of results from a query until the follow a link

And users tend to scan from top to bottom as they examine the query results

So if a site doesn't make it to the first page of search results, it might as well not exist for most users

Search Engine Optimization Techniques

Techniques for improving the number and positioning of a site's listing in response to a web query involve:

- On page factors -- doing things to a page's content
- Off-page factors -- adding links to the site from other pages

The META Tag Specification: HTML 4.01 (12/99)

```
<!ELEMENT META - O EMPTY  -- generic metainformation -->
<!ATTLIST META
  %i18n;      -- lang, dir, for use with content --
  http-equiv  NAME  #IMPLIED  -- HTTP response header name  --
  name        NAME  #IMPLIED  -- metainformation name --
  content     CDATA  #REQUIRED -- associated information --
  scheme      CDATA #IMPLIED  -- select form of content --
  >
```

What the W3C imagined:

```
<META NAME="DESCRIPTION" CONTENT="accurate prose description">
<META NAME="KEYWORDS" CONTENT="useful comma-separated keywords">
```

META tag: The Reality

November 2005 advice from www.highrankings.com:

"If this META tag were a child, it would be put into a foster home due to all the abuse it has received over the years"

... a bit disingenuous, perhaps, given that in 2000 (thanks, archive.org) this same firm said:

Our years of experience promoting our clients' websites in the search engines have proven that designing a website with keywords in mind, is the best way to ensure that the website is found by highly targeted users

Readings for 12/1 -- Multimedia Search and Retrieval

Alejandro Jaimes, Mike Christel, Sbastien Gilles, Ramesh Sarukkai, and Wei-Ying Ma, "Multimedia Information Retrieval: What is it, and why isnt anyone using it?"