信息组织与检索

第10讲：XML检索

主讲人：张蓉
华东师范大学 数据科学与工程学院

提纲

1. 上一讲回顾
2. 简介
3. 基本的XML概念
4. XML IR中的挑战
5. 基于向量空间模型的XML IR
6. XML IR评价

提纲

1. 上一讲回顾
2. 简介
3. 基本的XML概念
4. XML IR中的挑战
5. 基于向量空间模型的XML IR
6. XML IR评价

上一讲内容

- 交互式相关反馈(Interactive relevance feedback): 在初始检索结果的基础上，通过用户交互指定哪些文档相关或不相关，然后改进检索的结果

- 最著名的相关反馈方法： Rocchio 相关反馈

- 查询扩展(Query expansion): 通过在查询中加入同义或者相关的词项来提高检索结果
  - 相关词项的来源: 人工编辑的同义词词典、自动构造的同义词词典、查询日志等等。

相关反馈的基本思想

- 用户提交一个(简短的)查询
- 搜索引擎返回一系列文档
- 用户将部分返回文档标记为相关的，将部分文档标记为不相关的
- 搜索引擎根据标记结果计算得到信息需求的一个新查询表示。当然我们希望该表示好于初始的查询表示
- 搜索引擎对新查询进行处理，返回新结果
- 新结果可望（理想上说）有更高的召回率

Rocchio 1971 算法 (SMART系统使用)

实际中使用的公式:

$$\begin{aligned}\vec{q}_m &= \alpha\vec{q}_0 + \beta\mu(D_r) - \gamma\mu(D_{nr}) \\ &= \alpha\vec{q}_0 + \beta\frac{1}{|D_r|}\sum_{\vec{d}_j\in D_r}\vec{d}_j - \gamma\frac{1}{|D_{nr}|}\sum_{\vec{d}_j\in D_{nr}}\vec{d}_j\end{aligned}$$

q_m: 修改后的查询; q_0: 原始查询;

D_r、D_{nr}: 已知的相关和不相关文档集合

α, β, γ: 权重

- 新查询向相关文档靠拢而远离非相关文档
- α vs. β/γ 设置中的折中: 如果判定的文档数目很多，那么 β/γ可以考虑设置得大一些
- 一旦计算后出现负权重，那么将负权重都设为0
- 在向量空间模型中，权重为负是没有意义的。

伪相关反馈(Pseudo-relevance feedback)

- 伪相关反馈对于真实相关反馈的人工部分进行自动化
- 伪相关反馈算法
  - 对于用户查询返回有序的检索结果
  - 假定前 k 篇文档是相关的
  - 进行相关反馈 (如 Rocchio)
- 平均上效果不错
- 但是对于某些查询而言可能结果很差
- 几次循环之后可能会导致查询漂移(*query drift*)

查询扩展(Query expansion)

- 查询扩展是另一种提高召回率的方法
- 我们使用“全局查询扩展”来指那些“查询重构(query reformulation)的全局方法”
- 在全局查询扩展中，查询基于一些全局的资源进行修改，这些资源是与查询无关的
- 主要使用的信息: 同义词或近义词
- 同义词或近义词词典(thesaurus)
- 两种同(近)义词词典构建方法：人工构建和自动构建

查询扩展的类型

- 人工构建的同(近)义词词典 (人工编辑人员维护的词典，如 PubMed)
- 自动导出的同(近)义词词典 (比如，基于词语的共现统计信息)
- 基于查询日志挖掘出的查询等价类 (Web上很普遍，比如上面的 “palm” 例子)

本讲内容

- XML IR中的基本概念
- XML IR中的挑战
- XML IR中的向量空间模型
- XML IR评价

提纲

1. 上一讲回顾
2. 简介
3. 基本的XML概念
4. XML IR中的挑战
5. 基于向量空间模型的XML IR
6. XML IR评价

IR vs. 关系数据库

IR 系统常常与关系数据库(RDB)进行对比

- 传统上说，IR 系统从无结构文本(*unstructured text*, 指没有标记的“生”文本--“raw” text without markup)中返回信息
- RDB系统主要用于查询关系型数据(*relational data*)，即一系列记录集合，这些记录中包含预先定义的属性及属性值，如员工号、职位和工资

	RDB搜索	非结构化检索	结构化检索
对象	记录	非结构化文档	以文本为叶节点的树
模型	关系模型	向量空间或其他	?
主要数据结构	表格	倒排索引	?
查询语言	SQL查询	自由文本查询	?

一些包含文本的结构化数据最好建模成结构化文档而不是关系型数据，结构化文档的检索称为结构化检索 （structured retrieval）

结构化检索(Structured retrieval)

基本配置： 结构化或非结构化查询+结构化文档

结构化检索的应用场景

数字图书馆、专利数据库、博客、包含已标注命名实体（如人名、地名）的文本

例子

- 数字图书馆: *give me a full-length article on fast fourier transforms*
- 专利: *give me patens whose claims mention RSA public key encryption and that cite US patent 4,405,829*
- 实体标记文本: *give me articles about sightseeing tours of the Vatican and the Coliseum*

超凡CHOFN®
一站式知识产权解决方案专业提供商

Name: *

Number: *

Email:

QQ Number:

Description: *

(*) is required

submit

RDB并不适合上述场景

采用RDB会存在下列三个主要问题

1. 无序的DB系统可能返回大量文章，这些文章提到 Vatican、the Coliseum和sightseeing tours，但是并没有按照它们和查询的相关度排序
2. 大部分用户都很难精确描述结构化的限制条件。比如，用户可能并不知道搜索系统支持对哪些结构化元素的查询

   tours AND (COUNTRY: *Vatican* OR

   LANDMARK: *Coliseum*)?

   tours AND (STATE: *Vatican* OR BUILDING: *Coliseum*)?
3. 用户可能对结构化搜索和高级搜索很不熟悉，或者他们压根就不想用这些搜索功能。

解决办法：将排序检索模型用于结构化文档搜索来解决上述问题

结构化检索

	RDB搜索	非结构化检索	结构化检索
对象	记录	非结构化文档	以文本为叶节点的树
模型	关系模型	向量空间或其他	?
主要数据结构	表格	倒排索引	?
查询语言	SQL查询	自由文本查询	?

一个对结构化文档进行编码的标准：Extensible Markup Language (XML)

- 结构化IR → XML IR
- 同样适用于其他标记语言 (HTML, SGML, …)

提纲

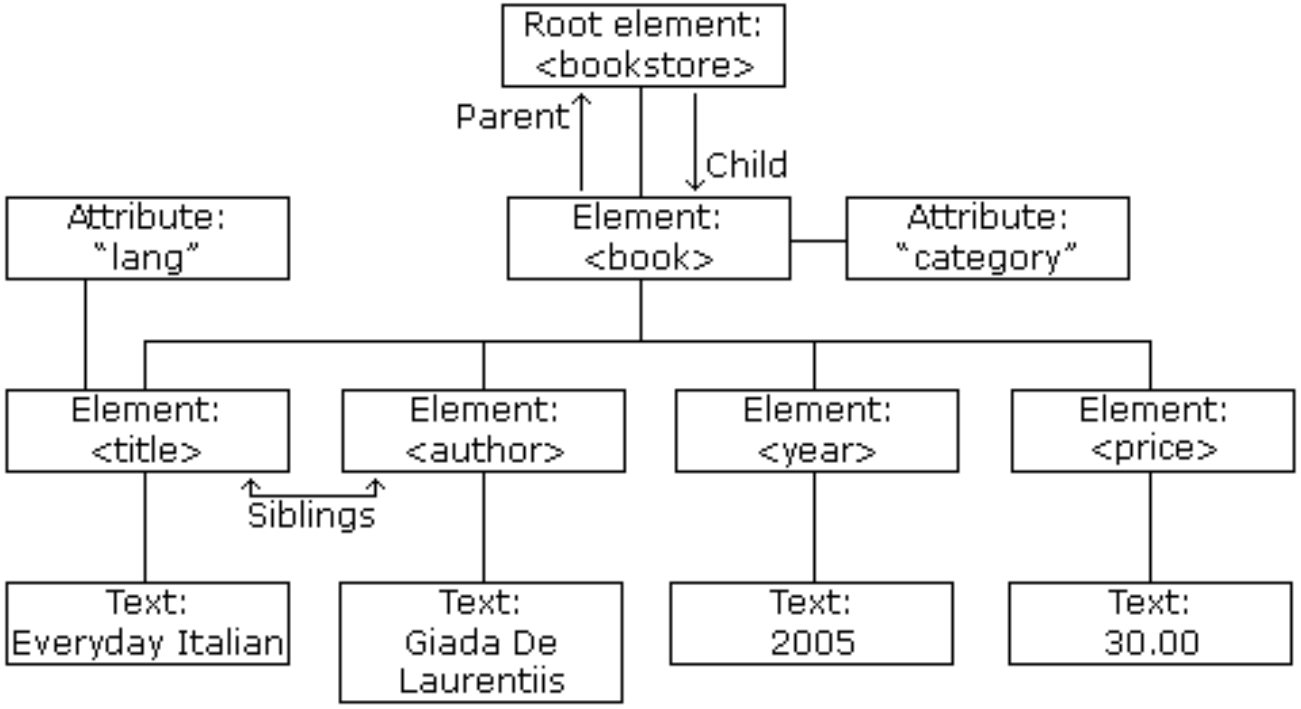
1. 上一讲回顾
2. 简介
3. 基本的XML概念
4. XML IR中的挑战
5. 基于向量空间模型的XML IR
6. XML IR评价

XML 文档

- 有序带标记树
- 树上的每个节点都是一个XML元素（XML element），它由起始标签（tag）和结束标签来界定(如 <title…>, </title…>)
- 一个XML元素可以有一个或多个XML属性（XML attribute）。(如 number)
- 属性可以有属性值 (如 vii)
- 属性可以有子元素 (如 title, verse)

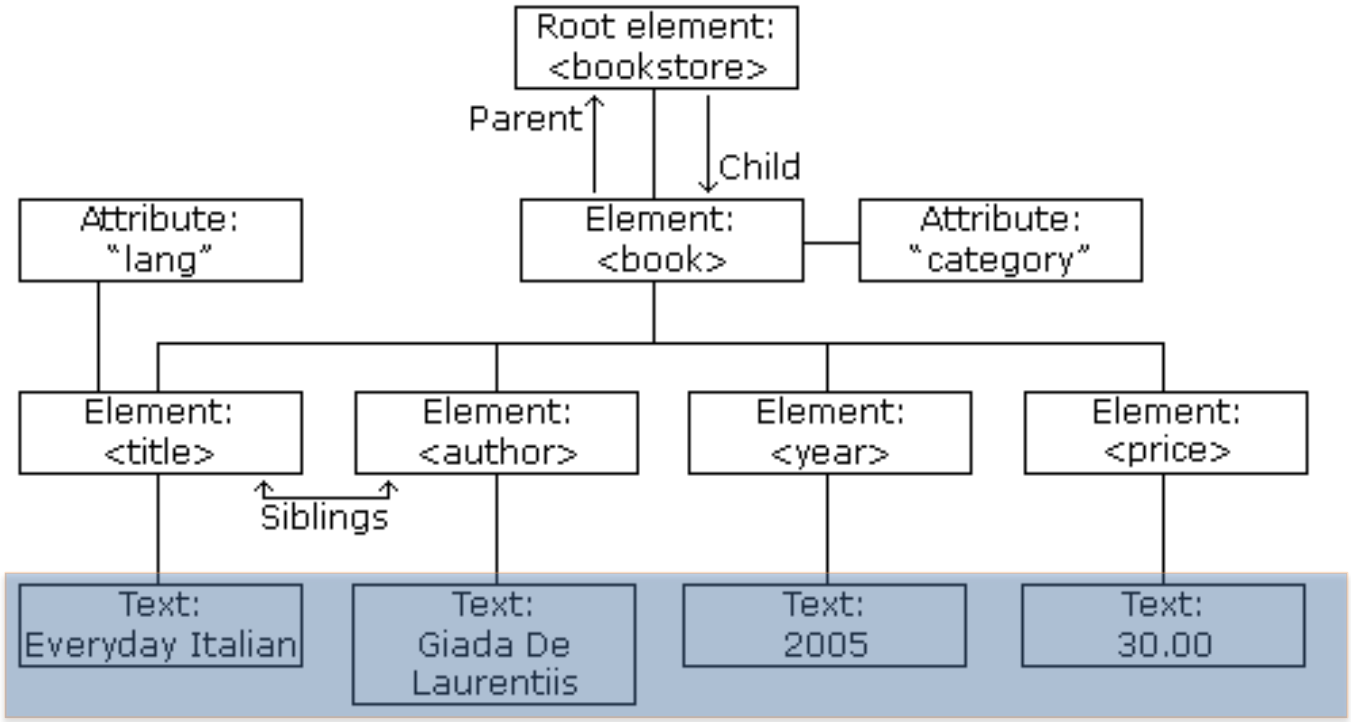
```
< bookstore >
  < book  category ="COOKING" >
    < title  lang ="en" > Everyday Italian </ title >
    < author > Giada De Laurentiis </ author >
    < year > 2005 </ year >
    < price > 30.00 </ price >
  </ book >
  < book  category ="CHILDREN" >
    < title  lang ="en" > Harry Potter </ title >
    < author > J K. Rowling </ author >
    < year > 2005 </ year >
    < price > 29.99 </ price >
  </ book >
  < book  category ="WEB" >
    < title  lang ="en" > Learning XML </ title >
    < author > Erik T. Ray </ author >
    < year > 2003 </ year >
    < price > 39.95 </ price >
  </ book >
</ bookstore >
```

```
< bookstore >
  < book  category ="COOKING" >
    < title  lang ="en" > Everyday Italian </ title >
    < author > Giada De Laurentiis </ author >
    < year > 2005 </ year >
    < price > 30.00 </ price >
  </ book >
  < book  category ="CHILDREN" >
    < title  lang ="en" > Harry Potter </ title >
    < author > J K. Rowling </ author >
    < year > 2005 </ year >
    < price > 29.99 </ price >
  </ book >
  < book  category ="WEB" >
    < title  lang ="en" > Learning XM
    < author > Erik T. Ray </ author
    < year > 2003 </ year >
    < price > 39.95 </ price >
  </ book >
</ bookstore >
```



XML 文档

叶节点由文本构成



XML 基础知识

- **XML 文档对象模型(XML Documents Object Model , XML DOM)**: 访问和处理XML文档的标准
  - DOM将元素、属性及元素内部的文本表示成树中的节点
  - 使用DOM API，我们可以从根元素出发，依据元素间的父子关系，自上而下对整个XML文档进行处理

- **XPath**: XML文档集中的路径表达式描述标准
  - 路径表达式也称为XML上下文(context)或直接称上下文

- **Schema**: 给出了XML文档所允许的结构限制条件。比如，莎士比亚剧本的schema规定，场（scene）只能以幕（act）的子节点方式出现。
  - XML文档的两个schema标准分别是 XML DTD （document type definition，文档类型定义）和 XML schema

提纲

1. 上一讲回顾
2. 简介
3. 基本的XML概念
4. XML IR中的挑战
5. 基于向量空间模型的XML IR
6. XML IR评价

挑战1: 返回文档的一部分

XML检索中，用户希望返回文档的一部分（即 XML元素），而不像非结构化检索那样往往返回整个文档

例子

如果在《莎士比亚全集》中查找Macbeth's castle，那么到底应该返回场（scene）、幕（act）还是整个剧本呢？

- 上述情况下，用户可能在查找 场(scene)
- 但是，另一个没有具体指定返回节点的查询Macbeth，应该返回剧本的名称而不是某个子单位

解决办法: 结构化文档检索原理(structured document retrieval principle)

23

课堂提问

- 结构查询举例

结构化文档检索原理

结构化文档检索原理

选择最合适的文档部分:
系统应该总是检索出回答查询的最明确最具体的文档部分

- 上述原理会引发这样一种检索策略，即返回包含信息需求的最小单位
- 但是，要在算法上实现这种原理是非常困难的。比如查询：title:*Macbeth*，整个剧本的标题*Maccbeth*以及第一幕第六场的标题*Macbeth's castle*都是包含匹配词项Macbeth的较好的命中结果。
  - 然而在这个例子中，剧本的标题这个位于更高层的节点作为答案却更合适
  - 确定查询应答的正确层次是非常困难的。

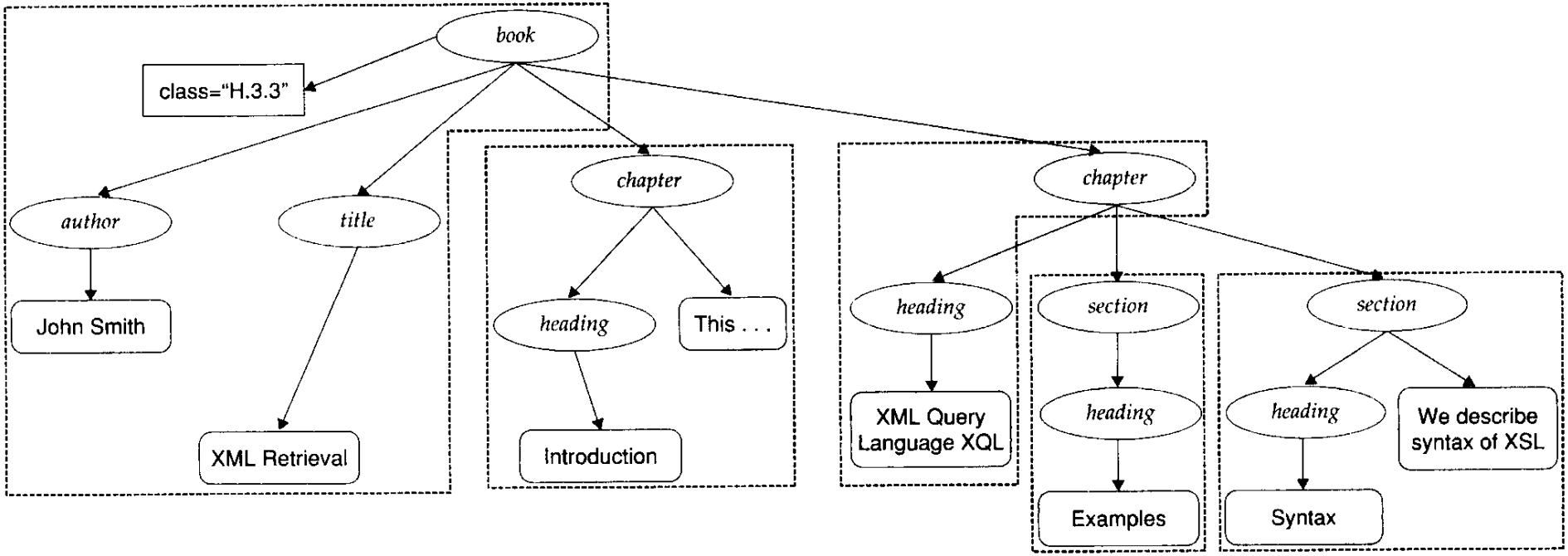
挑战2: 如何确定文档的索引单位

IR索引和排名中的核心概念：文档单位或索引单位

- 在非结构化检索中，合适的文档单位往往比较明显，如PC上的文档、邮件、Web上的网页等等
- 而在结构化检索中，却有定义索引单位的一系列不同的方法
  1. 将节点分组，形成多个互不重叠的伪文档（pseudodocument）
  2. 索引最大元素，然后自顶向下(top down)后处理
  3. 索引叶节点，然后自底向上(bottom up)进行后处理扩展
  4. 对所有元素建立索引

XML索引单位: 方法1-不重叠伪文档法

将节点分组，形成多个互不重叠的伪文档



索引单位: book、chapter、section但是这些单位之间没有重叠

缺点: 由于这些伪文档的内容不连贯，所以它们对用户而言可能没有意义

XML索引单位：方法2-自顶向下法

自顶向下(分两步走):

1. 可以使用最大的一个元素作为索引单位，比如，在一个书的集合中以元素book为索引单位
2. 然后通过对结果进行后处理，从而在每本书中找到更适合作为答案的子元素

上述”分两步走”的做法往往并不能返回最佳匹配子元素，这是因为整本书与查询的相关性并不能代表其子元素与查询的相关性

XML索引单位：方法3-自底向上法

自底向上:

与上述先返回大单位然后识别子元素（自顶向下）的方法不同的是，可以直接对所有叶节点进行搜索并返回最相关的一些节点，然后通过后处理将它们扩展成更大的单位

和自顶向下存在类似的问题：单个叶节点的相关性往往不能代表包含该叶节点的上层元素的相关性

XML索引单位: 方法4—索引所有元素

索引所有元素是限制最少的方法，也存在下列问题:

- 很多XML元素并不是有意义的搜索结果，比如，排版相关的元素（如definitely）或者不能脱离上下文进行单独解释的ISBN书号等
- 对所有元素建立索引也意味着搜索结果将会有高度的冗余性。

例子

查询Macbeth's castle，我们会返回从根节点到Macbeth's castle路径上的所有play、act、scene和title元素。此时，叶节点在结果集合中会出现4次，其中1次是作为索引对象直接出现的，而其他3次是作为其他元素的一部分出现的

We call elements that are contained within each other **nested elements.**元素之间互相包含的关系称为嵌套（nested）。对用户而言，在返回结果中包含冗余的嵌套元素则显得不太友好

挑战3：元素嵌套

针对元素嵌套所造成的冗余性，普遍的做法是对返回元素进行限制。这些限制策略包括： 这些限制策略包括：

- 忽略所有的小元素
- 忽略用户不会浏览的所有元素类型（这需要记录当前XML检索系统的运行日志信息
- 忽略通常被评估者判定为不相关性的元素类型（如果有相关性判定的话）
- 只保留系统设计人员或图书馆员认定为有用的检索结果所对应的元素类型

在大部分上述方法中，结果集中仍然包含嵌套元素。

挑战3：元素嵌套

进一步的处理策略:

- 可以通过一个后处理过程来去掉某些元素，从而降低索引结果的冗余性
- 在结果列表中将多个嵌套元素折叠起来，并通过对查询词项进行高亮显示（highlighting）来吸引用户关注相关段落

高亮显示

- 好处 1: 允许用户扫描中等规模的元素 (如 section)，因此，如果section和paragraph都出现在结果列表中，那么显示section就足够了。
- 好处2: paragraph会和它的上下文（包含该paragraph的section）在一起展示。即使paragraph本身就可以满足查询的需求，这种上下文仍然对于解释该paragraph很有帮助。

嵌套元素和词项统计信息

与嵌套相关的另一个挑战：在计算用于排序的词项统计信息(特别是idf)时，需要区别词项的不同上下文

例子

Gates出现在author节点下与其出现在内容元素中（如section中，此时代表的是gate的复数）毫无关系。于是，在这个例子中，为Gates计算一个单独的文档频率df意义不大。

解决办法：为每一个XML上下文-词项对计算idf

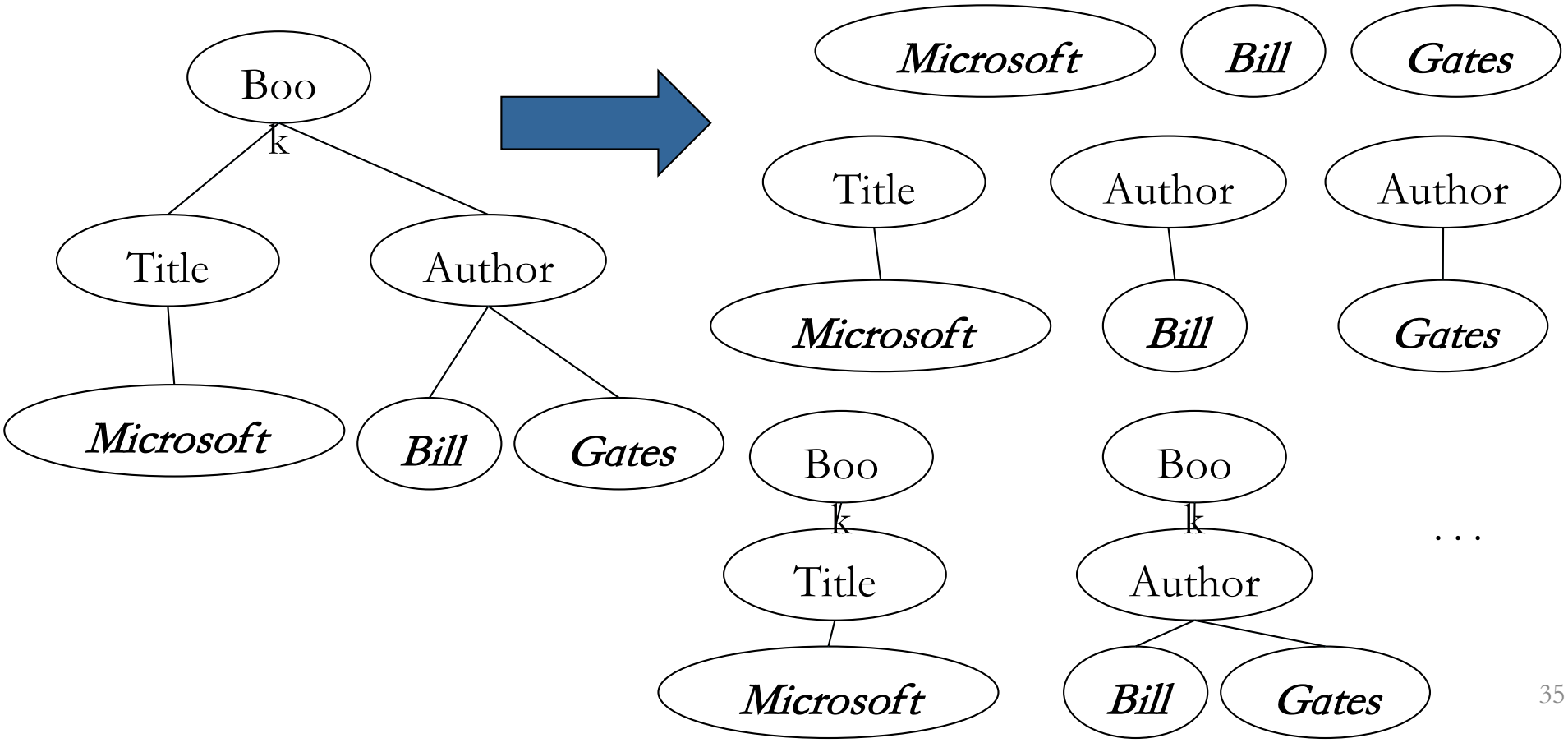
- 数据稀疏问题 (许多上下文—词项对出现过少从而导致对文档频率估计的可靠性不足)
- 一个折中方案是在区分上下文时只考虑词项的父节点x，而不考虑从根节点到x路径上的其他部分。

提纲

1. 上一讲回顾
2. 简介
3. 基本的XML概念
4. XML IR中的挑战
5. 基于向量空间模型的XML IR
6. XML IR评价

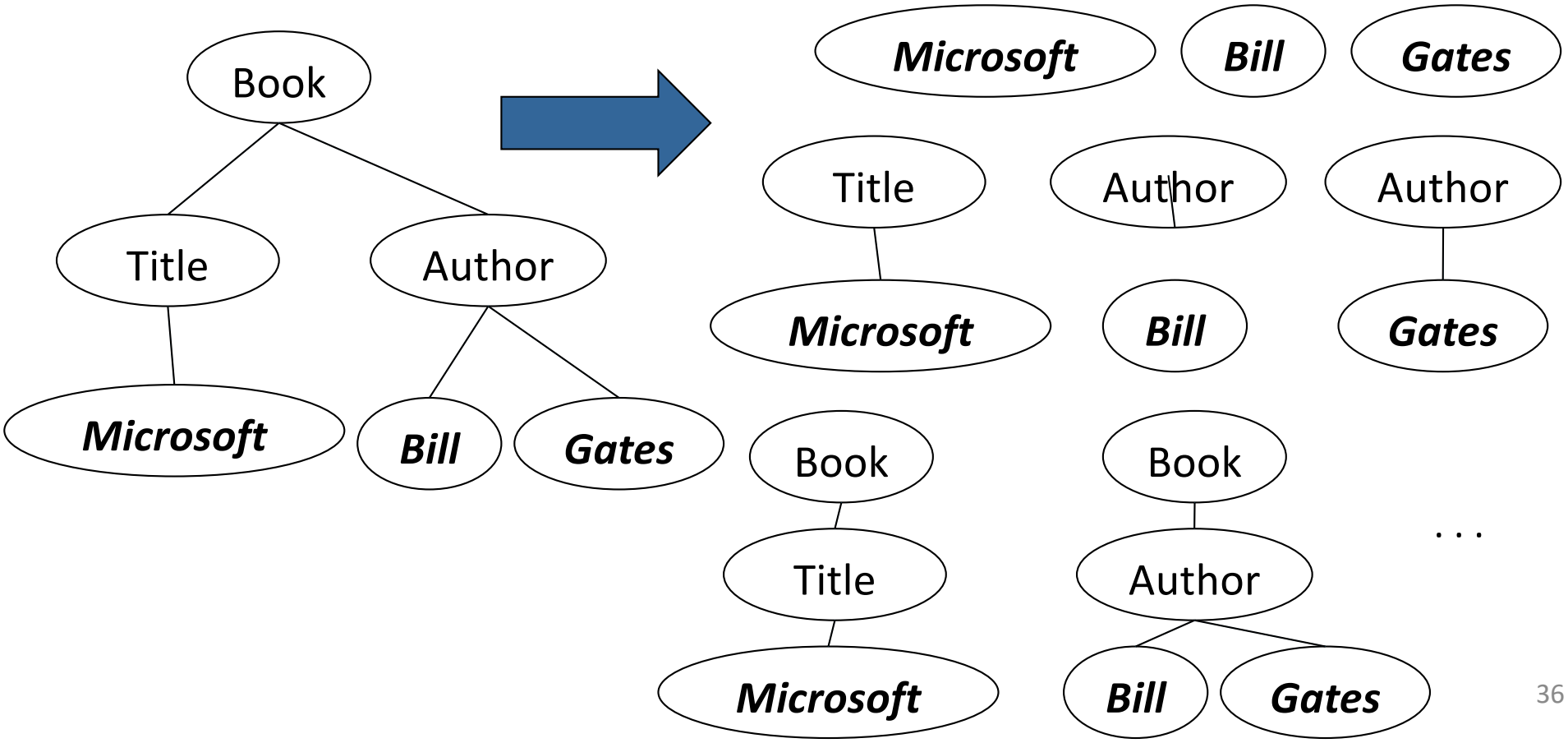
主要思路: 词汇化子树(lexicalized subtree)

目标: 对向量空间中的每一维都同时考虑单词及其在XML树中的位置信息
做法: 将XML文档映射成词汇化子树



主要思路: 词汇化子树

1. 考虑每个文本节点（叶节点）并将它们分裂成多个节点，每个节点对应一个词。例如，将 *Bill Gates* 分裂成 *Bill* 和 *Gates*
2. 我们将向量空间的每一维定义为文档的词汇化子树，这些子树至少包含词汇表中的一个词项



词汇化子树

于是就可以将查询和文档表示成这些词汇化子树空间上的向量，并根据前面的向量相似度公式进行相似度计算

非结构化 vs. 结构化IR中的向量空间相似度计算方法

两者的主要区别在于，前者中向量的每一维对应一个词项，而后者中对应一棵词汇化子树

结构化词项(Structural term)

在向量空间维度和查询的精度之间存在着一个折中

- 如果将每一维限制为词汇表中的词项，那么得到 的是一个标准的向量空间检索系统。这种系统下得到的很多文档在结构上并不与查询匹配 (例如，title中的 *Gates和author中的Gates*)
- 如果每棵词汇化子树都对应空间一维，那么空间的维数会变得太大

折中方案:对所有的最终以单个词项结束的路径建立索引，换句话说，对所有的XML上下文/词项对建立索引。这种XML上下文/词项对被称为结构化词项（structural term），记为 <*c*, *t*>:其中*c*是XML上下文，*t*是词项

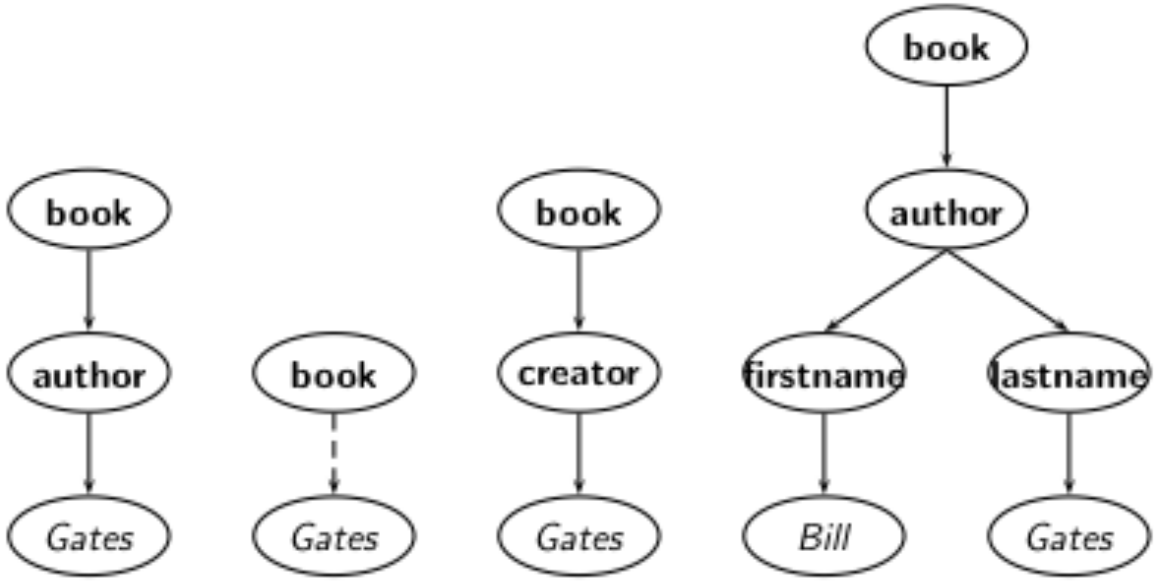
上下文相似度(Context resemblance)

一个简单的度量查询中路径c_q和文档中路径c_d相似度的指标是上下文相似度（context resemblance）函数C_R，其定义如下：

$$C_R(c_q,c_d)=\begin{cases}\dfrac{1+|c_q|}{1+|c_d|} & \text{如果}c_q\text{和}c_d\text{匹配}\\ 0 & \text{如果}c_q\text{和}c_d\text{不匹配}\end{cases}$$

其中$|c_q|$和$|c_d|$分别是查询路径和文档路径中的节点数目，并且当且仅当可以通过插入额外的节点使c_q转换成c_d时，c_q和c_d才能匹配。

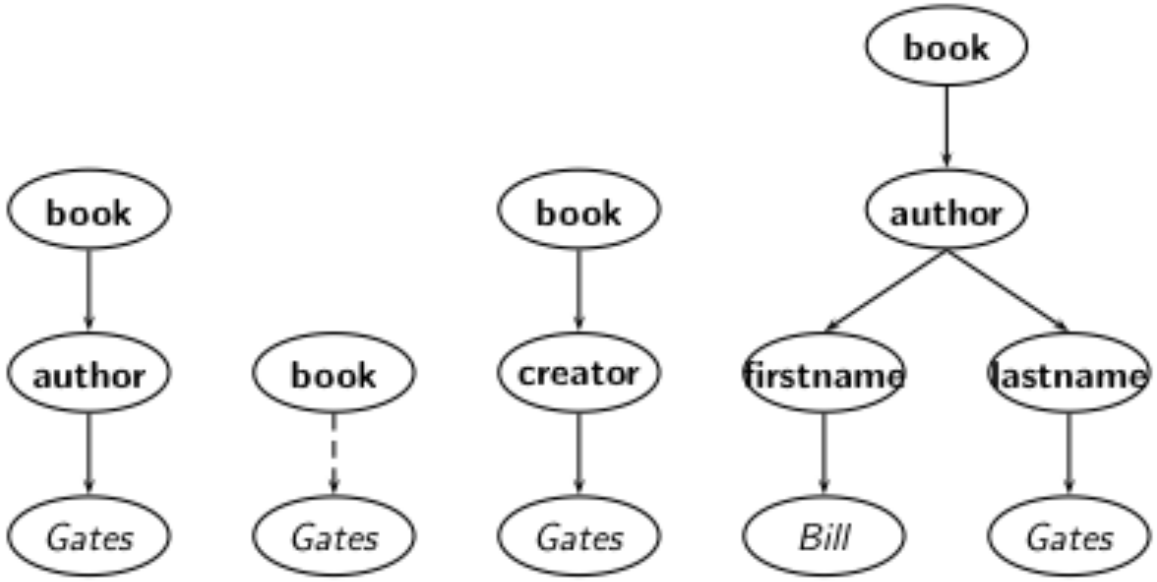
上下文相似度计算的例子



$$C_R(c_q, c_d) = \begin{cases} \dfrac{1+|c_q|}{1+|c_d|} & \text{如果} c_q \text{和} c_d \text{匹配} \\ 0 & \text{如果} c_q \text{和} c_d \text{不匹配} \end{cases}$$

$C_R(c_{q4}, c_{d2}) = 3/4 = 0.75$. 如果$q$和$d$相等，那么 $C_R(c_q, c_d) = 1.0$

上下文相似度计算的例子



$$C_R(c_q, c_d) = \begin{cases} \dfrac{1+|c_q|}{1+|c_d|} & \text{如果} c_q \text{和} c_d \text{匹配} \\ 0 & \text{如果} c_q \text{和} c_d \text{不匹配} \end{cases}$$

$C_R(c_{q4}, c_{d3}) = 3/5 = 0.6$.

提纲

1. 上一讲回顾
2. 简介
3. 基本的XML概念
4. XML IR中的挑战
5. 基于向量空间模型的XML IR
6. XML IR评价

INEX(Initiative for the Evaluation of XML retrieval)

INEX: XML检索研究中的首要评测平台，它通过协作产生参考文档集、查询集及相关性判断。在每年一度的INEX会议上，研究人员展示并讨论交流各自的研究结果。INEX 2002文档集包含大概12000篇来自IEEE期刊的文章。
(自2006 年开始，INEX使用英文Wikipedia这个更大的库)
文档的相关性判定主要通过人工判断来完成

INEX 2002 文档集统计信息	
12,107	文档数目
494 MB	规模
1995—2002	文章发表年份
1,532	平均每篇文档中的XML节点个数
6.9	平均每个节点的深度
30	CAS主题的数目
30	CO 主题的数目

INEX 主题

两种类型:

1. 仅基于内容(content-only 或 CO)的主题:和非结构化信息检索中一样的常规关键词查询
2. 内容结构相结合 (content-and-structure 或 CAS)的主题 : CAS 主题在关键词基础上增加了结构化限制

由于CAS查询同时包含结构信息和内容信息，其相关性判断就比非结构化中的相关性判断要复杂得多

INEX相关性判断

INEX 2002定义了部件覆盖度(Component coverage)和主题相关性(topical relevance)作为相关性判断的两个方面

部件覆盖度

评价的是返回元素在结构上是否正确，也就是说，其在树中的层次既不太高也不太低。

有四种情况:

1. 精确覆盖（E）。所需求的信息是部件的主要主题，并且该部件是一个有意义的信息单位
2. 覆盖度太小（S）。所需求的信息是部件的主要主题，但是该部件不是一个有意义（自包含）的信息单位
3. 覆盖度太大（L）。所需求的信息在部件中，但不是主要主题
4. 无覆盖（N）。所需求的信息不是部件的主题

INEX相关性判断

主题相关性也有4个层次：强相关（3）、较相关（2）、弱相关（1）和不相关（0）

部件覆盖度和主题相关性的组合

每个部件在覆盖度和主题相关性两个方面都要进行判断，然后将判断结果组合成一个数字—字母编码,2S表示一个比较相关的部件，但是其覆盖度太小。而3E表示高度相关并具有精确覆盖的一个部件。理论上说，4个等级的覆盖度和4个等级的相关性相组合，则对一个部件的评价有16种可能，但是实际评价中很多组合并不会出现。比如，一个不相关的部件不可能具有精确覆盖度，所以，编码为3N的组合是不可能的。

INEX 相关性判断

相关度—覆盖度组合可以采用如下量化方法：

$$\mathbf{Q}(rel, cov) = \begin{cases} 1.00 & \text{if} \quad (rel, cov) = 3\text{E} \\ 0.75 & \text{if} \quad (rel, cov) \in \{2\text{E}, 3\text{L}\} \\ 0.50 & \text{if} \quad (rel, cov) \in \{1\text{E}, 2\text{L}, 2\text{S}\} \\ 0.25 & \text{if} \quad (rel, cov) \in \{1\text{S}, 1\text{L}\} \\ 0.00 & \text{if} \quad (rel, cov) = 0\text{N} \end{cases}$$

传统的非结构化检索的二值相关性判断对于XML检索来说是不合适的。一个2S部件提供的信息尽管不完整，而且如果没有更多的上下文将很难进行解释，但是它却能部分地回答查询。量化函数Q并不强制使用二值相关性（相关或不相关），而是通过对部件划分等级来处理部分相关性。

于是，检索结果集合A中相关部件的数目可以定义为：

$$\#(\text{relevant items retrieved}) = \sum_{c \in A} \mathbf{Q}(rel(c), cov(c))$$

INEX 的评价指标

非结构化IR中定义的正确率、召回率及*F*值的标准定义都可以近似地应用到上式所示的相关部件数目上来。和前面的细微差别是，这里计算的是评分等级之和而不是二值相关性之和

缺点

没有考虑到重合现象，搜索结果中的元素的多重嵌套问题加剧了这一点

近年来INEX的焦点：提出结果无冗余的检索算法和评估指标并对结果进行恰当评价

本讲小结

- XML IR中的基本概念
- XML IR中的几大挑战
- XML IR中的向量空间模型：词汇化子树
- XML IR评价：INEX评测

参考资料

- 《信息检索导论》第10章

- Amer-Yahia, Sihem, and Mounia Lalmas. 2006. XML search: Languages, INEX and scoring. *SIGMOD Record* 35(4):16-23. DOI: http://doi.acm.org/10.1145/1228268. 1228271
- Harold, Elliotte Rusty, and Scott W. Means. 2004. *XML in a Nutshell*, 3rd edition. O'Reilly
- INEX网站: http://www.inex.otago.ac.nz/

致谢

- 本课件参考中科院计算所王斌老师的课件