清 华 大 学

综 合 论 文 训 练

题目：大数据环境下信息抽取模板自动聚类与发现

系　　别：计算机科学与技术系

专　　业：计算机科学与技术

姓　　名：丘骏鹏

指导教师：朱小燕

辅导教师：郝宇

2013 年 6 月 8 日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签　名：_____ 导师签名：_____ 日　期：_____

中文摘要

论文的摘要是对论文研究内容和成果的高度概括。摘要应对论文所研究的问题及其研究目的进行描述，对研究方法和过程进行简单介绍，对研究成果和所得结论进行概括。摘要应具有独立性和自明性，其内容应包含与论文全文同等量的主要信息。使读者即使不阅读全文，通过摘要就能了解论文的总体内容和主要成果。

论文摘要的书写应力求精确、简明。切忌写成对论文书写内容进行提要的形式，尤其要避免“第 1 章……；第 2 章……；……”这种或类似的陈述方式。

本文介绍清华大学论文模板 ThuThesis 的使用方法。本模板符合学校的本科、硕士、博士论文格式要求。

本文的创新点主要有：

- 用例子来解释模板的使用方法；
- 用废话来填充无关紧要的部分；
- 一边学习摸索一边编写新代码。

关键词是为了文献标引工作、用以表示全文主要内容信息的单词或术语。关键词不超过 5 个，每个关键词中间用分号分隔。（模板作者注：关键词分隔符不用考虑，模板会自动处理。英文关键词同理。）

关键词： TEX；LATEX；CJK；模板；论文

ABSTRACT

An abstract of a dissertation is a summary and extraction of research work and contributions. Included in an abstract should be description of research topic and research objective, brief introduction to methodology and research process, and summarization of conclusion and contributions of the research. An abstract should be characterized by independence and clarity and carry identical information with the dissertation. It should be such that the general idea and major contributions of the dissertation are conveyed without reading the dissertation.

An abstract should be concise and to the point. It is a misunderstanding to make an abstract an outline of the dissertation and words "the first chapter", "the second chapter" and the like should be avoided in the abstract.

Key words are terms used in a dissertation for indexing, reflecting core information of the dissertation. An abstract may contain a maximum of 5 key words, with semicolons used in between to separate one another.

Keywords: TEX; LATEX; CJK; template; thesis

目 录

主要符号对照表

HPC	高性能计算 (High Performance Computing)
cluster	集群
Itanium	安腾
SMP	对称多处理
API	应用程序编程接口
PI	聚酰亚胺
MPI	聚酰亚胺模型化合物，N-苯基邻苯酰亚胺
PBI	聚苯并咪唑
MPBI	聚苯并咪唑模型化合物，N-苯基苯并咪唑
PY	聚吡咙
PMDA-BDA	均苯四酸二酐与联苯四胺合成的聚吡咙薄膜
ΔG	活化自由能 (Activation Free Energy)
χ	传输系数 (Transmission Coefficient)
E	能量
m	质量
c	光速
P	概率
T	时间
v	速度
劝学	君子曰：学不可以已。青，取之于蓝，而青于蓝；冰，水为之，而寒于水。木直中绳。（车柔）以为轮，其曲中规。虽有槁暴，不复挺者，（车柔）使之然也。故木受绳则直，金就砺则利，君子博学而日参省乎己，则知明而行无过矣。吾尝终日而思矣，不如须臾之所学也；吾尝（足齐）而望矣，不如登高之博见也。登高而招，臂非加长也，而见者远；顺风而呼，声非加疾也，而闻者彰。假舆马者，非利足也，而致千里；假舟楫者，非能水也，而绝江河，君子生非异也，善假于物也。积土成山，风雨兴焉；积水成渊，蛟龙生焉；积善成德，而神明自得，圣心

备焉。故不积跬步，无以至千里；不积小流，无以成江海。骐骥一跃，不能十步；驽马十驾，功在不舍。锲而舍之，朽木不折；锲而不舍，金石可镂。蚓无爪牙之利，筋骨之强，上食埃土，下饮黄泉，用心一也。蟹六跪而二螯，非蛇鳝之穴无可寄托者，用心躁也。—— 荀况

第 1 章　引言

1.1　研究背景

1.1.1　大数据研究背景

随着互联网的快速发展，互联网上的信息呈现了爆发式的增长，互联网已经成为人们获取信息的一个主要渠道。举个例子，2011 年底，新浪微博注册用户数超过 3 亿，每日发微博量超过 1 亿[1]，在 2012 年底用户数更是突破了 5 亿[2]。到 2015 年, 将会有近 30 亿人在使用互联网，产生和共享的数据将达到 8ZB① [3]。随着我们可以获得的数据量的不断增加，人们的研究工作也受到了新的挑战。传统的数据处理手段正愈发显示出其局限性，如何有效对海量的数据进行处理，进而挖掘出我们所需要的内容逐渐成为一个重要的问题。近几年，“大数据”迅速成为计算机科学领域非常受关注研究方向。

Doug Laney 在[4] 中，提出了“大数据”的 3 个特点：容量（volume）、速度（velocity）和多样性（variety）。容量是指数据的存储量非常大，通常在 TB，甚至 PB 级别；速度是指数据的产生速度很快；多样性是指产生的数据多种多样，没有一个固定的类型，大部分都是以非结构化和半结构化数据的形式存在。这些是我们在面对大数据时所需要解决的主要问题。

之前数据挖掘方面许多研究，更多地是关注如何在有限数据的情况下尽可能多地提取出准确的我们关心的信息。由于受到数据量的限制，很多数据中隐藏的模式和信息并不能被有效地发现。如今，海量的数据使得数据本身不再是我们研究中的瓶颈，我们关注的重点更多的在于如何从这些有大量重复冗余的数据中找到我们真正关心的那部分信息，将信息提取出来，组成结构化的信息，用于计算机的后续处理。

① 1 ZB = 10^{21}B

1.1.2 结构化数据简介

从结构上来看，数据可以分为非结构化数据，半结构化数据和结构化数据。结构化数据是指可通过明确的结构，如表或者树的格式，进行统一表示的数据。关系数据库和定义良好的 XML 就是存储结构化数据的两个典型例子。非结构化的数据没有统一的格式，比如各种各样的文本、图像、声音、视频等。半结构化的数据则有一定的结构，但结构并不固定，有些字段可能会扩充或者删除。目前人们日常所接触的大部分万维网上的信息，大部分都是通过 HTML 文档进行表示的，HTML 文档就是一种典型的半结构化数据，不同的文档在结构上可能有很大的变化。可以看出，非结构化数据和半结构化的数据更适用于人机界面的交互，而结构化数据则对机器更加友好。为了便于用计算机进行存储和后续处理，在用计算机处理各种各样的数据的时候，我们常常希望能将其他的非结构化或者半结构化的数据转化成结构化的数据进行表示。

这篇文章中，我们主要关心的对象是半结构化的 HTML 文档。为了更好地对 HTML 文档进行处理，我们需要将 HTML 文档中我们关心的信息提取出来，用结构化数据的方式（比如 XML）进行存储。例如，对于我们获得的博客数据，我们主要关心其中的标题，正文和评论的信息，那么可以建立一个 XML 文档，其中的字段都是固定的，每个文档对应一个 `document` 节点，`document` 节点下面有 `author`,`content` 和 `comment` 三个子节点。我们将每个博客的 HTML 中对应部分抽取出来，存储到该 XML 文档中。如图 1.1所示。

1.1.3 HTML 文档的模板

互联网上有成千上万的 HTML 文档，对于大部分的网站来说，不可能针对每一个网页单独写一个静态网页存储到服务器。实际上，我们在互联网上浏览的大部分 HTML 网页都是通过网站的后台程序动态生成的，只有极少量的还是通过静态 HTML 方式进行存储。

在 Web 开发领域，MVC(Model, View, Controller) 模式是目前最流行的开发方法。模型（Model）是对底层数据和业务的进行的封装；视图（View）负责用户界面的交互，包括给用户发送信息，接受用户输入等；控制器（Controller）则是系统的控制逻辑，对用户请求进行处理，用选择合适的视图用于显示模型返回的数据。如图 1.2所示[①] 。目前有很多基于 MVC 模式的 Web 开发框架，比如

① 来源：http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

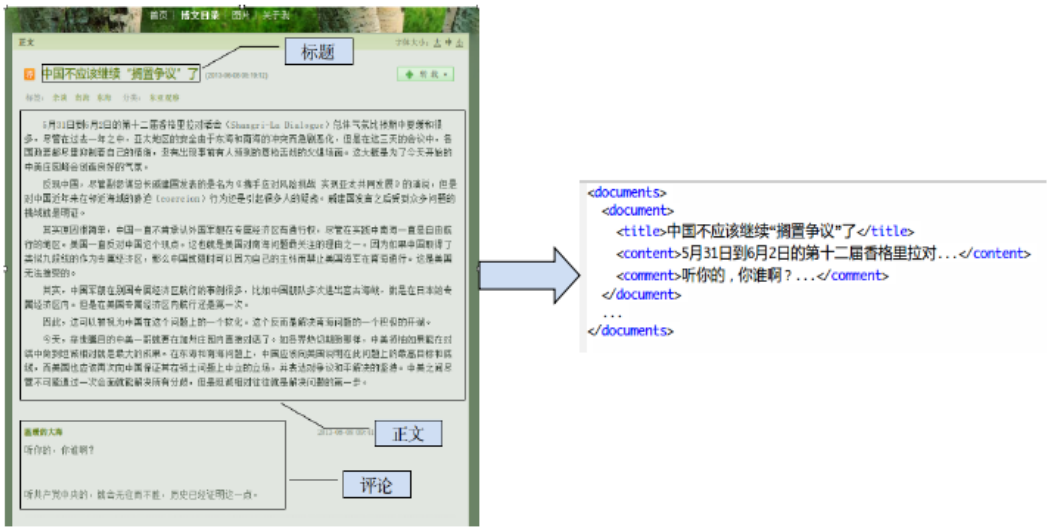

图 1.1　用 XML 存储博客的正文，标题和评论

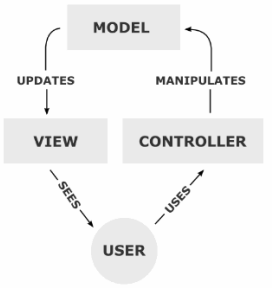

图 1.2　MVC 模式

```
<html>
  <body>
    <h1>{{ news.title }}</h1>
    <p>{{ news.content }}</p>
  </body>
</html>
```

图 1.3　Django 中的 HTML 模板示例

基于 Ruby 语言的 Ruby on Rails，基于 Python 语言的 Django 以及基于 Scala 语言的 Play! Framework 等等。这些框架用模型对底层的数据库进行包装，当用户请求一个 HTML 页面的时候，控制器负责从数据库中查询相应数据，然后在视图层选择合适的渲染模板，将对应的数据填充到模板中，从而生成目标 HTML 文档，将其返回给用户。图 1.3是一个简单的用 Django 开发网站时视图层的模板示例，其中 `news.title` 和 `news.content` 是从数据库中得到的查询结果。

我们注意到，大部分 HTML 网页都是通过这种查询底层数据库得到相应数据，然后使用模板进行渲染的方法生成的。对于大量的从同一个模板生成的网页来说，“模板”就是这些网页的“公共部分”。实际上，模板的定义要比这里所说的网页的“公共部分”要复杂一些，许多 Web 开发框架的模板引擎都支持

```
<html>
  <body>
    <h1>{{ news.title }}</h1>
    <p>{{ news.content }}</p>
    <div>
      {% for comment in news.comments %}
        <li>{{ comment }}</li>
      {% endfor %}
    </div>
  </body>
</html>
```

图 1.4　使用了 for 的 Django 模板示例

一些简单的控制结构（如 if，for 等），如果生成模板的时候使用了这些控制逻辑，对应的模板也会比较复杂，如图 1.4所示。我们在 5中将会给出一个较为严格的定义，这里我们先使用这种直观的定义便于理解。如果我们能够从大量的由同一个模板生成的网页中将它们的模板抽取出来，那么我们就可以用抽取出来的模板对其他的由同一个模板生成的网页进行内容的抽取。简单来说，有了网页的模板以后，每个网页中非“模板”的部分即可以认为是通过查询后台数据库得到的数据动态生成的，而这些这是我们所需要的。

1.2　相关工作

1.3　本文重点难点和主要内容

1.4　本章总结

第 2 章　系统框架

2.1　整体框架介绍

2.2　网页过滤模块

2.3　网页聚类模块

2.4　模板生成模块

2.5　本章小结

第 3 章　基于后缀树的重复记录检测

3.1　后缀树简介

3.2　Ukkonen 后缀树构建算法

3.3　重复记录检测算法

3.4　本章小结

第 4 章　网页相似度计算与聚类

4.1　基于最长公共子串的网页距离计算

4.2　算法优化与改进

4.3　聚类算法实现

4.4　本章总结

第 5 章　模板生成和内容提取

5.1　模板生成

5.2　内容提取

5.3　本章总结

第 6 章　系统实现和实验结果

6.1　系统具体实现

6.2　实验数据和实验环境

6.3　实验结果

6.4　结果分析和存在的问题

第 7 章　工作总结和未来展望

7.1　工作总结

7.2　未来工作展望

[5]

插图索引

表格索引

参考文献

[1] 新华网. 新浪微博注册用户突破 3 亿每日发博量超过 1 亿条

[2] 新华每日电讯. 新浪微博用户注册数超 5 亿

[3] Kalakota R. Big Data Infographic and Gartner 2012 Top 10 Strategic Tech Trends, 2011. http://practicalanalytics.wordpress.com/2011/11/11/big-data-infographic-and-gartner-2012-top-10-strategic-tech-trends/

[4] Laney D. 3D Data Management: Controlling Data Volume, Velocity and Variety. Technical report, Gartner, February 06, 2001. http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf

[5] IEEE Std 1363-2000. IEEE Standard Specifications for Public-Key Cryptography. New York: IEEE, 2000

致　谢

衷心感谢导师 xxx 教授和物理系 xxx 副教授对本人的精心指导。他们的言传身教将使我终生受益。

在美国麻省理工学院化学系进行九个月的合作研究期间，承蒙 xxx 教授热心指导与帮助，不胜感激。感谢 xx 实验室主任 xx 教授，以及实验室全体老师和同学们的热情帮助和支持！本课题承蒙国家自然科学基金资助，特此致谢。

感谢 ThuThesis，它的存在让我的论文写作轻松自在了许多，让我的论文格式规整漂亮了许多。

声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____ 日　期：_____

附录 A　书面翻译

文档结构相似性算法调研

摘要

这篇论文对文档结构相似性算法做了简明的调研，包括优化的树编辑距离算法和各种近似算法。这些近似算法包括简单加权标签相似度算法，文档结构的傅里叶变换，和将连续序列技术应用到结构相似度计算上。我们展示了三个令人惊奇的结果。第一，傅里叶变换的方法是所有近似算法中最不精确的一个，同时也是最慢的一个。第二，优化的树编辑距离的算法并不一定是最好的用来将不同网站的网页进行聚类的算法。第三，对于许多应用来说，最简单的结构的近似可能是最有用也是最有效率的机制。

A.1　简介

随着万维网上大量的文档的出现，自动地处理这些文档，将其应用于信息抽取，近似聚类和搜索的需求越来越大。在这个领域的主要工作主要集中在文档的内容上。然而，虽然万维网的继续发展和进化，越来越多的信息被放在结构化的富文本中，从 HTML 转换到 XML。这个结构化的信息是一个文档意义的重要体现。从文档中辨别出结构上“相似”，或者结构上互相“包含”的那些文档的方法是一个非常重要那些相关的相似文档关联起来的机制，而这些文档可能包含不同的文本内容，那些基于文本内容的相似度算法起不到这样的作用。

现在已有几个文档结构在其中起到关键作用的应用。目前的信息提取算法隐式或显式地依赖文档的结构化元素。结构化的信息能帮助我们将大量的从不同网站上获得的网页整理成一些可以大致可以比较的集合。这就使得软件能够将可以抽取出正确结果的集合和那些不能抽取有用信息的集合分离开来。

结构相似度是一个非常重要的话题，有非常多的算法可以计算任意两个文档结构之间的最小编辑距离。然而，由于这些算法复杂度非常高，通常都需要

$O(n^2)$ 或者更多的时间去计算距离，因此创造一些更快的，但是距离的计算精确度有些许损失的算法是有可能的。

我们在这篇论文的第二节展示了当前用于检测结构相似度的算法的概述。之后在第三节我们描述了一个新的基于连续序列的计算结构相似度的近似算法。在第四节我们在速度和精确度上对比了一下这些近似度算法和优化的树编辑距离算法。在第五节中我们用对不同算法特点的概括进行了总结。

A.2 当前研究状况

基于树的文档结构性相似度的研究已经有很长的历史了。早期基于树的变换检测来自 [1]，[2]。近期 Shasha [3]，[4]，[5] 和 Chawathe [6]，[7] 做了一些关于树到树变换的研究，主要集中在如何创建一个最小的脚本用来进行树之间的转换。在将一些基于结构的相似度计算修改成半结构的格式方面也有很多的尝试，包括 NiagraCQ [8]，Xdiff [9]，适用于 XML 的 Xyleme [10]，[11]，以及 AIDE [12]，[13] 和适用于 HTML 的 ChangeDetector™。

之前关于 HTML 文档相似度的工作大部分集中在了内容相似度上，页面分段 [15] 技术也是一样。目前的结构相似度集中在了 XML 的 Schema 的相似度上。DTD 相似度研究集中在对成对的文档和未知但相似的 DTD 的近似度计算上。这要求每两篇文档比较一次需要 $O(n^2)$ 的计算复杂度。其他的工作将文档之间结构相似度的问题转化为用傅里叶变换的时间序列的相似度，实现上采用快速傅里叶变换以实现 $O(n \times lgn)$ 的复杂度。

在这篇论文中，我们引进了一个将连续序列技术应用在衡量文档结构相似度上的方法。这要求用 $O(k)$ 的复杂度去创造一个连续序列文档（在这里 k 表示节点的个数），以及提供常数时间的复杂度进行文档之间的比较。在计算时间上的节省是通过损失计算精确度的得到的，实际可以任意减小精确度以满足不同的要求。第四节比较了这个技术与其他近似算法在不同的数据集上的表现。

A.2.1 近似算法

这里我们提供一个对不同类型的用于计算文档相似度算法的概述。我们将描述的衡量标准包括树编辑距离，标签距离，傅里叶变换和路径相似度。连续序列技术的动机和算法我们将在第三节给出。

树编辑距离相似度。一些作者提供了一些计算两个树之间优化的树编辑距离的算法。这篇论文使用 Nierman 和 Jagadish [16] 描述的动态编程实现。一般来说，编辑距离衡量的是将一个树转换为另一个树所要求的插入，删除和更新的最少的节点个数。通过将编辑操作的次数和较大的那个文档的节点个数之间进行归一化，可以将其转换为相似度的衡量标准。令 N_i 为文档 D_i 的树形表示的节点集合，于是

$$TED(D_i, D_j) = \frac{editDistance(D_i, D_j)}{max(|N_i|, |N_j|)}$$

标签相似度。标签相似度可能是结构相似度最简单的度量方法，因为它只衡量两个标签集合之间的近似程度。在 XML 文档中，标签是 schema 的一个组成部分，使用一个类似标签集合的页面 schema 也很有可能类似。两个文档的标签集合可以用来测量它们的重合程度。令 T_i 为文档 D_i 所包含的标签集合，T_J 为文档 D_j 包含的标签集合。两个页面的简单的标签相似度为 T_i 和 T_j 的交集和并集的比例。

然而，由于各种原因，这并不令人满意。一个关键的问题是遵循一个相同 schema 的页面，比如 HTML，只有一个非常有限的不同标签的个数；一个页面可能包含非常多的一个特殊的标签，但是用于比较的页面只包含相对很少的这个标签。为了补偿标签频率，我们可以引入一个加权的相似度度量。令 t_{ik} 为 T_i 的一个成员，w_{ik} 为标签 t_{ik} 在文档 D_i 中出现的次数。此外，令 t_{jk} 为 T_j 中对应的标签，v_{jk} 为 t_{jk} 在文档 D_j 中出现的次数。如果有 n 个互不相同的标签同时出现在文档 D_i 和 D_j 中，加权标签相似度可以表示为：

$$WTS(D_i, D_j) = \frac{\sum_{k=1}^{n} 2 * min(w_{ik}, v_{jk})}{\sum_{k=1}^{n} (w_{ik} + v_{jk})}$$

由于这个仅包含每个文档中的标签集合，结构相似度的精确性与使用的标签无关。因此，这个度量方法在类似 HTML 的环境中应该只有非常低的精确度，这些环境标签集合是很有限的，但是结构却变化非常大。它在一些都遵循一个小的 schema 集合的文档上有可能变得更加精确，因为这些 schema 限制了文档结构的变化。

傅里叶变换相似度衡量。Flesca *et al.* 引入傅里叶变换作为计算文档相似度的一个手段。基本的想法是将一个文档的开始标签和结束标签以外的全部信息去掉，只保留一个表示结构的骨干。然后将这个结构转化为一个数字序列。将

这个数字序列视为时序序列，然后使用傅里叶变换将其转化到频域。最终，两个文档之间的距离可以通过两个信号的幅度差别进行计算。

这个算法有几个对结果有重大影响的关键组成部分。像一开始所说的，文档结构的编码采用一个独特的（序列的）正数来表示每一个开始标签，用负数来表示对应的结束标签。属性被当做标签。注意到这个意味着为了比较两个文档，标签的数字映射必须在每个文档中都是一样的。Flesca *et al.* 选择了一个文档 d 的多层的编码，将其映射成一个序列 $[S_0, S_1, \ldots, S_n]$，其中

$$S_i = \gamma(t_i) \times expF(t_i) + \sum_{t_j \in nest_d(t_i)} \gamma(t_j) \times expF(t_j)$$

其中 $\gamma(t_i)$ 是一个对应于第 i^{th} 个标签的整数，$expF(t_i) = B^{maxdepth(D)-l_{t_i}}$ 是一个决定标签权重的指数因子，B 是一个固定的底，$maxdepth(D)$ 是正在比较的文档的最大深度，l_{t_i} 是第 i^{th} 个标签的深度，$nest_d(t_i)$ 是 t_i 的祖先集合。

最终的两个文档 d_1, d_2 的距离度量通过傅里叶变换定义为

$$dist(d_1, d_2) = (\sum_{k=1}^{M/2} (|[FFT(h_1)](k)| - |[FFT(h_2)](k)|)^2)^{\frac{1}{2}}$$

其中 FFT 傅里叶变换关于同时出现在 h_1 和 h_2 中频率的插值，h_1 是 d_1 对应的信号，M 是出现在插值中的点的个数。

这个方法有一些困难的地方。第一，FFT 要求点的个数是 2 的幂。一个 DFT 实现使用了和文档的时序表示一样多的点，意味着 DFT 和 FFT 近似算法的精度是不一样的。对于我们的比较而言，我们发现了 FFT 当成 DFT 时是 $O(n^2)$ 的，在实践中它比树编辑距离算法更慢，而且 DFT 的精度比 FFT 的精度要低。

第二，调和标签映射的要求和预先计算用于比较文档的最大深度意味着需要有根据单一文档来预先计算这个算法的任意部分的能力，以减小成对比较所需的时间。

第三，傅里叶变换典型的用法是在重复的无限时间序列上。我们所碰到的那些文档是有限的。为了可以使用这个变换，我们必须将从文档中提取的时间序列扩展为无限重复的。对于原始文档这意味着什么以及这个对于比较有那些影响我们都不清楚。

A.3 路径连续序列

使用优化的树编辑距离算法的问题在于它在大的文档集上代价非常大。目前所展示的近似算法要么不够直观（傅里叶变换），要么只提供了一个粗略的相似度度量（加权标签算法）。我们感到需要有一个可以根据应用来调整精度的快速的近似算法。

Shingle 是 Broder 在 [15] 中为计算文本文件相似度和包含度而引入的技术。这个技术减小了文档中词或者标记的集合，将其变成一个散列的列表，能够直接和另一个文档进行比较，通过集合的差集，并集和交集来计算相似度或者包含度。

更进一步地，shingle 集合的子集，叫做 sketch，可以用于计算文档的相似度。直观地说，sketch 是一个页面的随机采样的一段文本。关键在于由于随机映射在所有的页面中都是一样的，而且结果是排好序的，这些样本是可以直接在不同的页面之间进行比较。页面采样的重合度意味着整个页面的重合程度。

我们将展示如何修改这个技术，将其应用到文档的结构中。这使我们可以在线性时间计算文档结构相似度。通过稍微减小精度，可以实现常数时间的任意大小的文档之间的比较。

A.3.1 路径相似度

为了创建一个适用 shingle 技术的结构编码，我们必须找出一个方法去创建一个标记的列表，用于表示这个结构。一些自然的选择，比如使用深度优先或者广度优先的堆编码，因为用于表示树的标记列表本身的不分段特性从而证明是不合适的。这意味着当一个分支结束和另一个分支开始的时候，覆盖序列的窗口不能被辨别出来。覆盖这些的窗口不能精确表示原始文档的任一部分。为了找到用于这种编码的合适的分段方法，衍生了另一个自然的编码方法：路径。

半结构化的文档（HTML 和 XML）可以被看成一个由分支和从根到叶的路径组成的序列。为了我们的目标，我们将任意的部分路径，即从根到文档中任意的节点组成的路径也认为是一个标记。一个树可以用一个这些标记的序列来表示。比如，HTML 中最简单的树有一个 title 和 body 元素，可以被编码为：

/html

/html/head

/html/head/title/

/html/head/title/[text]
/html/body /html/body/[text]

路径相似度衡量两个不同文档之间的路径的相似度。一个路径被定义为一个从根节点开始，到叶子节点结束的相连的节点列表。路径相似度可以从多种方式进行度量：二元的，路径要么相等或者不相等；部分的，在每个路径中可以比较的节点的数目是知道的；或者加权的，节点根据它到根的距离进行加权。

部分路径相似度需要很大的计算代价。因为两个树的路径有 $n!$ 种可能的映射，用穷举的算法来产生优化的相似度分数是不可行的。二元的相似度代价要小很多，因为在一个版本中的每一个独特的路径可以通过数据库的 join 技术来匹配另一个版本中的等价路径。相似度可以通过匹配上的路径个数和两个树总的路径数的比例来计算。

A.3.2 将 shingle 应用到路径上

一个 shingle 是一个从文档中得到的连续的标记的子序列。两个文档的相似性定义为

$$r(D_i, D_j) = \frac{S(D_i, w) \cap S(D_j, w)}{S(D_i, w) \cup S(D_j, w)}$$

然而 $S(D_i, w)$ 是从文档 D_i 创建长度为 w 的 shingle 的操作符。类似地，文档 D_i 和 D_j 的包含度定义为

$$c(D_i, D_j) = \frac{S(D_i, w) \cap S(D_j, w)}{S(D_i, w)}$$

为了方便和快速的处理，shingle 可能通过散列函数转换成数字。这个散列函数应该有较高的可信度使得两个 shingle 散列成一个数的冲突的可能性变得很小。让散列的空间显著大于 shingle 的空间可以使得构造这样一个合适的散列函数变得容易很多。依据一个 shingle 中标记的个数（或者窗口的长度），这可能会非常微妙。

一个将结构化比较的复杂度减小到 $O(1)$ 的关键技术是每个文档只保留一个相对小的 sketch。从 [15] 中我们可以看到从一个 shingle 集合的排列中进行随机采样得到的样本可以被当成两个文档之间相似度的无偏估算子。一个有效的实现这个的办法是通过伪随机数生成算法来散列值，将结果排序，然后选择最小的 k 的结果。

为了将 shingle 应用到路径结构中，我们定义 $S(D_i, w)$ 如下：对于 D_i 的树形表示中的每个节点，计算从根到该节点的路径；根据这个（部分）路径的标签名列表创建一个散列；将该散列加入到当前窗口；窗口向前滑动一个单位（即将窗口的第一个散列值去掉）。。

注意到根据定义，shingle 集合可能是一个集合或是一个包——类似于标签相似度和加权标签相似度之间的差别。于是就出现了使用一个集合来包含 shingle 会显著减小 shingle 的表达能力，并将更大的误差引入到估计中。

我们测量了路径 shingle 和部分路径 shingle 的精度。比较是在使用无穷的 k 的不同的窗口大小下进行的。用来比较是数据是从 `my.yahoo.com` 下载的在两年时间内的典型的网页快照。结果显示小的窗口大小（从 1 到 4）对精度没有影响：两个聚类都没有错误，四个聚类只引入百分之三的错误。用来比较的 shingle 个数的不同并不影响聚类的效果。k 的值从 10 测试到了 1000, 以及一个无穷数量的 shingle。所有的都显示了错误在百分之十之内。

A.4 实验

在这一节中我们经验性地评价不同的近似算法与树编辑距离相比的精度，同时还比较了不同的近似算法的性能。

所有的实验都是在一个配置了速龙双核 2G 赫兹的处理器，2.4Linux 内核的工作站上运行。算法使用 Java 实现的，在 Sun 1.4.2 JVM 上执行。所有的算法都采用了页摘要数据结构来实现，该数据结果比 DOM 树表示有很大的性能提升。性能的测量是取 10 次运行的平均值。

根据聚类进行比较。聚类是在树编辑距离的基础上用来衡量不同度量之间的效果的。树编辑距离是可证明的一个两个树之间的最佳的编辑距离。我们假定这是最好的相似度度量。其他的算法会产生一些不同的距离度量，不能直接和一个编辑距离进行比较。然而，如果一个大的文档集合是根据某个度量方法来聚类的，这些通过不同的度量方法产生的聚类是可以通过给定同一个聚类方法进行比较的。换句话说，如果两个文档通过树编辑距离判断是相似的，那么其他的度量也应该认为这两个文档是相似的，或者恰恰相反。

我们可以将一个在一个近似算法中被放在一个 shingle 聚类中，而在树编辑距离算法中放到另一个类中的归类成错误文档。这个错误度量有一些缺点。比

如，一个文档集合被分成两个聚类，任何度量都会有严格小于 50% 的最大错误率。一般地，随着聚类数目 n 的增长，最大错误率严格小于 $1 - \frac{1}{n}$。

我们用两个数据集进行聚类。第一个是从 500 个 XML 文档中综合生成的集合。这个集合建模一个书仓库，每个文档都列出一个书对应的作者，发行商和发行时期。文档之间唯一的结构差别在于这本书的作者的个数。

每个度量方法都用来度量两个文本之间距离。文档根据这些距离通过开 *k – means* 进行聚类。通过每种度量得到的聚类与树编辑距离得到的聚类进行比较，计算一个错误估计。结果显示在 Table I 中。

TABLE I

CLUSTER ERROR RATE OVER BOOK DATA

Similarity Metric	Error Rate 6 clusters	4 clusters	2 clusters
Weighted Tag	6%	5%	2%
FFT	60%	46%	47%
Path	0 %	0%	2%
Path Shingles	6%	5%	2%

我们测试了最多 6 个聚类，因为这个数据集中根据书籍的作者数量，有 6 个自然的聚类。我们期望加权标签度量有较低的错误率，因为这些数据之间的结构差异只有 `author` 这个标签出现的次数。

第二个数据集是一系列从以下网站得到的快照：`cnn.com`，`corona.bc.ca`，`news.gnome.org`，`10-10phonerates.com` 和 `my.yahoo.com`。快照是 2001 到 2003 两年期间的，大概是每天一次。冗余的快照（由 MD5 签名决定）被移除了，每个站点的快照集合取样 20 份页面。一样的聚类算法用在这些文档上，只有这次我们有一个预定义好的聚类（通过网站），并且可以将每个算法同这些预定义好的聚类进行比较。结果显示在 Table II 中。

TABLE II

CLUSTER ERROR RATE OVER WEB DATA

Similarity Metric	Error Rate 6 clusters
Weighted Tag	0%
Path	28%
Path Shingles	34%
TED	38%
FFT	45%

这个错误率好像异常地高，特别是对于我们在其他测试中用来当做基准的树编辑距离算法。我们推测这个错误率是由于 HTML 的结构中用来呈现给用户内容的词汇相对较小。路径和路径 shingle 度量比树编辑距离性能要好，但也比预期的要差。这可能是因为它们使用了部分路径来描述树的结构。很深的树在顶部区域可能会呈现很多相似的结构，导致两个来自不同站点的树的相似度可能会互相偏向。FFT 度量方法的糟糕的性能不能给出一个简单的解释。可以说依据相同的最开始的一批标签（`html`，`head` 和 `body`），每个网页都是极度相似的信号，在叶子层面也是相似的构造（标签列表和文本）。然而，这个转换使得我们难以找出是网页的哪一些特征导致它们变得如此相似。

作为最后的比较，我们检查了以上表格中从一个站点来的快照。这会在比如监测一个页面随着时间变化的时候很有用。我们选择 `my.yahoo.com` 因为里面的内容会定期发生变化，但是结构上随着时间的变化很缓慢（比如，当一个新的图片被加入到一些特点的假日）。我们再次把使用近似算法和树编辑距离产生的聚类进行比较。

TABLE III

CLUSTER ERROR RATE OVER YAHOO! DATA

Similarity Metric	Error Rate 6 clusters	4 clusters	2 clusters
FFT	33%	31%	29%
Path	59%	46%	1%
Weighted Tag	60%	50%	1%
Path Shingles	61%	47%	1%

我们观察到除了 FFT 以外的大部分近似算法在小的聚类大小上都有非常低的错误率，但是错误率在聚类的规模变大时会激增。观察一下描述树编辑距离聚类和 FFT 聚类以及路径/路径 shingle 聚类之间映射变化的矩阵是有意义的。Table IV 描述了 FFT 聚类和树编辑距离聚类的不同，Table V 描述了路径聚类和树编辑距离聚类之间的不同。

TABLE IV

CLUSTER COMPARISON BETWEEN FFT AND TED METRICS; 6 CLUSTERS

Cluster # FFT	TED 1	2	3	4	5	6
1	1	0	0	0	0	0
2	1	0	0	0	0	0
3	1	0	0	0	0	0
4	1	0	0	0	0	0
5	92	31	1	8	0	0
6	1	0	0	0	0	0

从这两个矩阵我们可以推导出 FFT 度量会倾向于将所有的数据点聚成一个类别，因此不提供一个有效的分辨相似网页的能力。在另一方面，路径度量

（路径 shingle 和加权标签度量实际是一样的）比树编辑距离更有分辨能力，能够比树编辑距离分出更好的类别。尽管这些实验中将这种情况标为了错误，它可能有很重要的分辨那些感觉上不一样，但在相同的编辑距离内的树的功能。

最惊奇的观察在于加权标签的相似度度量方法，一开始我们认为是一文不值的东西，却能和复杂很多的技术达到一样的精确程度。这可能可以归因为大部分实验是在来自一个站点的相对同构的页面上进行的（就像书籍数据集或者 Yahoo! 数据集的情况一样），或者可以归因为非同构的站点为了突出它们的对比而使用了不同 HTML 标签子集。另一个观察是傅里叶变换技术在很简单的数据集上表现也很差。这就使得我们认为尽管这是一个将结构转换为一个更简单的用于比较的格式的想法，但是它并不适用与文档结构，无论是从分析上还是实验上。

A.4.1 性能比较

选择一个近似算法的主要原因是为了将速度提高到一个可以接受的水平，因为最优的算法太慢了。文档聚类用于数据抽取或者搜索和获取方法，提供了一个结构化相似度的很好的例子。用于精度估计的相对小的数据集说明了近似算法在计算近似相似度上的有效性。Figure 1 展示了不同算法在书籍文档的数量对数增长时的相对代价。

聚类的时间是在一个小于一千比特的较小的文档上计算的。为了更好地理解计算两个文档之间差异的代价，我们比较了在 TCP-H 基准测试数据上的每个度量方法的代价。这个数据是由 Toxgene [19]XML 文档生成器随机生成的。我们修改了生成器的参数以便产生包含原始数据集合 1%，5%，10%，15%，20%，和 25% 的文档。这个生成器跑两次，在每个分数上产生两个不同的文档。结果在 Figure 2 中。

我们可以看到树编辑距离算法比任何其他的近似算法都要慢好几个数量级。FFT 算法也显示出了尽管它用很大的精度换来了速度，它仍然比加权标签或者路径 shingle 方法慢一个数量级，后两者的精度还显著更高。

大文件支持。以空间换时间的优化对于小到中型大小的文件都工作地很好，但是树编辑距离在其数据结构会超过物理内存的大文件上明显变慢。当物理内存被耗尽，机器被迫开始使用交换内存——这个会慢好几个数量级。

shingle 在创建常数大小的大文件指纹时有优势，消除了在计算时在内存中

维护复杂数据结构的需要。

shingle 还可以部分调优，就算是因此取了原始的指纹。给定一个窗口散列的集合，只有最前面的 k 个需要比较。用于比较的散列的数目可以调整，把精度用来换速度和空间。这就使得对于更低精度的比较，一次性可以有更多的指纹在内存中驻留。

A.5 总结

我们展示了一些测量文档结构相似度的算法，比较了它们的精度和性能。我们有了一些有趣的发现。

第一，我们提供了一个对 [17] 中所描述的傅里叶变换的实验性的批判。尽管相比最优化树编辑距离算法，傅里叶变换方法是一个更快的方法，但这个方法没有提供在不同情境中的一个精确的相似度衡量。此外，这个技术的性能也通常比其他更直观的相似度计算方法要差。

第二，对于很多结构相似度的应用来说，最简单的计算标签数量的方法提供了最好的性能情况下的一个可以接受的精度。我们一开始将加权标签相似度当成一个不重要的计算结构相似度最快的近似方法。然而，结果是它可以和其他任何近似算法表现一样好，甚至更好。尽管它没有像树编辑距离方法一样（匹配的相同子树），或者路径 shingle 方法一样（子结构包含性）提供一定的结构特征，但是对于不需要这些特征的应用来说，这个算法既快又有辨别力。

最后，我们基于文档结构中的路径展示了一个新的相似度度量。我们通过应用 shingle 技术，可以从任意的文档中创建常数大小的表示方法，使得比起其他相似度度量，聚类方法可以应用到大很多的文档集合中。此外，这种度量具有在一个大文档集合中搜索子结构和在基于树的文档集合中做一些结构化的挖掘的能力。

致谢

作者要感谢 Chunk Baldwin 和 Ghaleb Abdulla，他们刺激了这些话题之间的对话。作者还要感觉 Daniel Rocco，他搭建了第一个实验框架，并帮助开发了页面摘要数据结构，作为许多算法的基础。

这个工作是基于 No. W-7405- ENG-48. UCRL-CONF-202728 法令，在美国能源部的保护下，在 University of California Lawrence Livermore National Laboratory 进行的，

参考文献

[1] K. C. Tai, “The tree-to-tree correction problem,” Journal of the ACM, vol. 26, no. 3, 1979.

[2] S. Y. Lu, “A tree-to-tree distance and its application to cluster analysis,” IEEE Trans. Pattern Anal. Mach. Intell. PAMI-1, no. 2, 1979.

[3] D. Shasha and K. Zhang, “Fast algorithms for the unit cost editing distance between trees,” Journal of Algorithms, no. 11, 1990.

[4] K. Zhang, D. Shasha, and J. T.-L. Wang, “Approximate tree matching in the presence of variable length don’t cares,” J. Algorithms, vol. 16, no. 1, pp. 33–66, 1994. [Online]. Available: citeseer.nj.nec.com/zhang93approximate.html

[5] D. Shasha and K. Zhang, “Approximate tree pattern matching,” in Pattern Matching Algorithms. Oxford University Press, 1997, pp. 341–371. [Online]. Available: citeseer.nj.nec.com/95609.html

[6] S. Chawathe, A. Rajaraman, H. Garcia-Molina, and J. Widom, “Change detection in hierarchically structured information,” in Proceedings of ACM SIGMOD, 1996.

[7] S. S. Chawathe and H. Garcia-Molina, “Meaningful change detection in structured data,” in Proceedings of the 1997 ACM SIGMOD, 1997, pp. 26–37. [Online]. Available: citeseer.nj.nec.com/article/chawathe97meaningful.html

[8] J. Chen, D. DeWitt, F. Tian, and Y. Wang, “NiagaraCQ: A scalable continuous query system for Internet databases,” in Proceedings of the 2000 ACM SIGMOD, 2000.

[9] Y. Wang, D. DeWitt, and J.-Y. Cai, “X-Diff: An effective change detection algorithm for XML documents,” International Conference on Data Engineering, 2003.

[10] A. Marian, S. Abiteboul, G. Cobena, and L. Mignet, “Change- centric man-

agement of versions in an XML warehouse,” in The VLDB Journal, 2001, pp. 581–590. [Online]. Available: citeseer.nj.nec.com/marian00changecentric.html

[11] G. Cobena, S. Abiteboul, and A. Marian, “Detecting changes in XML documents,” in International Conference on Data Engineering, 2002, pp. 41 –52.

[12] F. Douglis, T. Ball, Y.-F. Chen, and E. Koutsofios, “The AT&T Internet difference engine: Tracking and viewing changes on the Web,” in World Wide Web, vol. 1, January 1998, pp. 27–44.

[13] Y.-F. Chen, F. Douglis, H. Huan, and K.-P. Vo, “TopBlend: An efficient implementation of HtmlDiff in Java,” in Proceedings of the WebNet2000 Conference, San Antonio, TX, November 2000.

[14] V. Boyapati, K. Chevrier, A. Finkel, N. Glance, T. Pierce, R. Stokton, and C. Whitmer, “ChangeDetector(TM): A site-level monitoring tool for the WWW,” in WWW2002, May 2002.

[15] A. Z. Broder, “On the Resemblance and Containment of Documents,” in Proceedings of Compression and Complexity of SEQUENCES 1997, 1997.

[16] A. Nierman and H. Jagadish, “Evaluating structural similarity in XML documents,” Fifth International Workshop on the Web and Databases, 2002.

[17] S. Flesca, G. Manco, E. Masciari, L. Pontieri, and A. Pugliese, “Detect- ing structural similarities between XML documents,” Fifth International Workshop on the Web and Databases, 2002.

[18] D. Rocco, D. Buttler, and L. Liu, “Page Digest for large-scale Web ser- vices,” in Proceedings of the IEEE Conference on Electronic Commerce, June 2003.

[19] D. Barbosa, A. O. Mendelzon, J. Keenleyside, and K. A. Lyons, “Toxgene: An extensible template-based data generator for XML,” in SIGMOD Conference, 2002. [Online]. Available: citeseer.nj.nec.com/525958.html

在学期间参加课题的研究成果

个人简历

xxxx 年 xx 月 xx 日出生于 xx 省 xx 县。

xxxx 年 9 月考入 xx 大学 xx 系 xx 专业，xxxx 年 7 月本科毕业并获得 xx 学士学位。

xxxx 年 9 月免试进入 xx 大学 xx 系攻读 xx 学位至今。

发表的学术论文

[1] Yang Y, Ren T L, Zhang L T, et al. Miniature microphone with silicon- based ferroelectric thin films. Integrated Ferroelectrics, 2003, 52:229-235. (SCI 收录, 检索号:758FZ.)

[2] 杨轶, 张宁欣, 任天令, 等. 硅基铁电微声学器件中薄膜残余应力的研究. 中国机械工程, 2005, 16(14):1289-1291. (EI 收录, 检索号:0534931 2907.)

[3] 杨轶, 张宁欣, 任天令, 等. 集成铁电器件中的关键工艺研究. 仪器仪表学报, 2003, 24(S4):192-193. (EI 源刊.)

[4] Yang Y, Ren T L, Zhu Y P, et al. PMUTs for handwriting recognition. In press. (已被 Integrated Ferroelectrics 录用. SCI 源刊.)

[5] Wu X M, Yang Y, Cai J, et al. Measurements of ferroelectric MEMS microphones. Integrated Ferroelectrics, 2005, 69:417-429. (SCI 收录, 检索号:896KM.)

[6] 贾泽, 杨轶, 陈兢, 等. 用于压电和电容微麦克风的体硅腐蚀相关研究. 压电与声光, 2006, 28(1):117-119. (EI 收录, 检索号:06129773469.)

[7] 伍晓明, 杨轶, 张宁欣, 等. 基于 MEMS 技术的集成铁电硅微麦克风. 中国集成电路, 2003, 53:59-61.

研究成果

[1] 任天令, 杨轶, 朱一平, 等. 硅基铁电微声学传感器畴极化区域控制和电极连

接的方法: 中国, CN1602118A. (中国专利公开号.)

[2] Ren T L, Yang Y, Zhu Y P, et al. Piezoelectric micro acoustic sensor based on ferroelectric materials: USA, No.11/215, 102. (美国发明专利申请号.)