基于HTML/CSS的网页结构相似性分析系统

杜雷鸣

一、设计概述

1.1 背景

在Web开发和设计中，网页结构的相似性分析可以帮助开发者识别和比较不同网站或网页的设计和布局模式，这种分析通常涉及对HTML和CSS代码的解析、向量化以及后序的相似度计算。

1.2 需求和设计调研

1.2.1 网页的表征方式

网页一般采用DOM树的表征方法，也有部分方案将网页表征为纯文本或图片的形式。纯文本和图片的表征方式仅在部分特殊场景下使用，解决单点的问题，有时候无法较为全面地表征结构化特征。而DOM树的表征方式由于其本身的“树”型结构，能比较好的反应网页的结构特征。

1.2.2 网页结构相似度计算方法

如1.2.1中所述，网页结构一般被表征为DOM树的结构，因此网页结构相似性的计算本质上是一个带权重（属性）的树之间的相似度计算问题，经过系统调研，一般有以下几种匹配算法：

- 基于树编辑距离的方法

算法原理：将两个网页的DOM树视为树结构，通过计算将一个DOM树转换为另一个DOM树所需的最小编辑操作数（如插入、删除、替换节点）来衡量它们之间的相似度。常见的如TED算法， FTM以及降低复杂度的SFTM算法等。

优点：能够捕捉到网页结构的细微差异，适用于复杂的网页结构比较。

缺点：算法复杂度较高，计算代价较大。

- 基于特征向量的方法

算法原理：将网页的DOM树结构表示为特征向量，通过比较特征向量之间的相似度来评估网页之间的结构相似度，例如对树的深度，宽度等类型特征进行特征抽取。

优点：算法简单直观，易于实现和理解；计算效率较高，适用于大规模网页集合。

缺点：需要选择合适的特征表示方法，可能无法捕捉到网页结构的信息。

- 基于子树核的方法

算法原理：将网页的DOM树结构表示为子树集合，通过比较子树集合之间的相似度来评估网页之间的结构相似度。

优点：能够捕捉到网页结构的局部相似性，适用于复杂的网页结构比较；可以通过选择合适的核函数来提高计算效率和准确性。

缺点：需要选择合适的子树表示方法和核函数，可能需要进行参数调优；对于大规模网页集合，计算复杂度较高，需要考虑优化算法以提高效率。

- 基于图匹配的方法

算法原理：将网页的DOM树表示为图结构，通过图匹配算法来比较两个网页之间的结构相似度。

优点：能够捕捉到网页结构的全局和局部相似性；可以使用图匹配算法来处理复杂的网页结构。

缺点：算法复杂度较高，计算效率较低；对于大规模网页，可能需要考虑优化算法以提高计算效率。

1.2.3 相似度的表征方式

常见的相似度计算方法有以下几种：

- 欧氏距离（Euclidean Distance）：欧氏距离是最常见的距离度量方法之一，用于测量向量之间的距离。它衡量了两个点之间的直线距离。在特征向量相似度匹配中，欧氏距离可以用来衡量两个特征向量之间的相似度。
- 余弦相似度（Cosine Similarity）：余弦相似度是一种度量两个非零向量之间的相似性的方法。它通过计算两个向量的夹角的余弦值来衡量它们之间的相似度。在文本分类、推荐系统等任务中经常使用余弦相似度来衡量文档或物品之间的相似度。

- 曼哈顿距离（Manhattan Distance）：曼哈顿距离是两个点在各个坐标轴上的距离总和。它也称为城市街区距离或 L1 距离。曼哈顿距离可以用于测量特征向量之间的距离，尤其在计算机视觉和图像处理中经常使用。
- 汉明距离（Hamming Distance）：汉明距离是衡量两个等长字符串在对应位置上不同字符的个数。在特征向量相似度匹配中，汉明距离通常用于比较二进制向量或文本数据。
- Jaccard 相似度（Jaccard Similarity）：Jaccard 相似度是两个集合交集大小与并集大小之比。在特征向量相似度匹配中，Jaccard 相似度通常用于比较集合数据，例如文档、商品或用户的集合。
- 皮尔逊相关系数（Pearson Correlation Coefficient）：皮尔逊相关系数衡量了两个变量之间的线性关系强度和方向。在特征向量相似度匹配中，皮尔逊相关系数可以用于衡量两个特征向量之间的相关性。

1.3 系统整体方案

如1.2中调研所述，结合需求中关于“向量化”的约束以及基于树和图的结构匹配算法复杂度都比较大的现实，本系统设计的整体方案（基于Python）如下：

1）使用requests库获取网页内容，使用BeautifulSoup库结合数据处理和预处理方案构建DOM树，此处单个DOM树节点包含节点的tag信息和处理后的attr信息（包含静态和动态CSS）。

2）使用特征向量 + 子树结构的算法进行特征抽取和构建，构建的特征需要包含节点信息和多重结构信息，使用基于hash函数(simhash)的方法进行节点信息的特征降维。

3）使用余弦相似度和Jaccard相似度两个指标进行树的特征向量的评估。

4）使用监督数据确定相似度阈值来判定是否相似。

1.4 系统概述

本系统旨在实现两个网页的结构相似性计算，因此需要包含基本的网页获取、数据处理和预处理（网页结构相关信息）、特征向量计算、相似度计算和相似度评估的功能。

本系统主要提供以下功能模块：

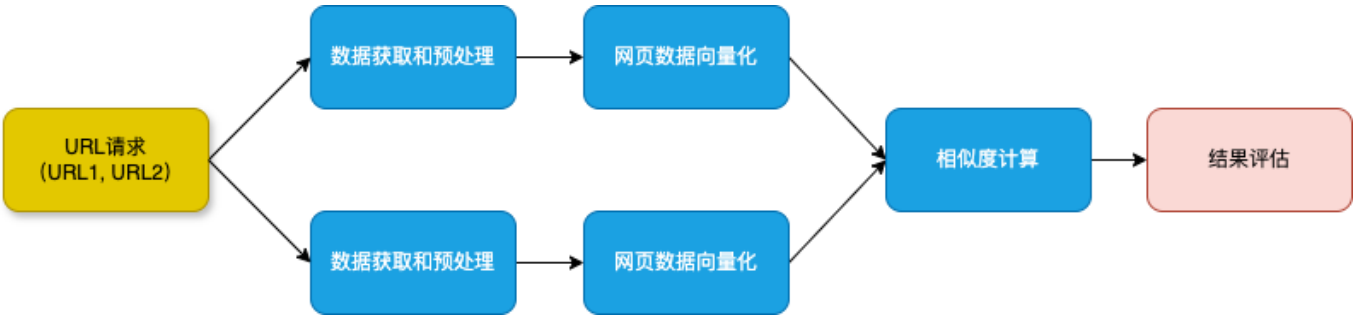
1）数据获取和预处理模块（包含网页获取，预处理，数据标准化等）

2）网页数据向量化模块（特征抽取、hash计算、映射、降维等）

3）相似度计算模块

4）结果评估模块

其基本处理逻辑如下图所示：



二、系统组成部分

2.1 数据获取和预处理模块

数据获取和预处理模块的主要功能是完成数据的获取和初步预处理，并将数据转换为合理的表达形式。

考虑到实际网页HTML文件的文本比较复杂，包含注释等可能与结构无关的信息模块，同时存在有HTML标记文本，CSS文本，JS文本以及大量的链接文本等，需要找到一种合适的网页数据结构表达方式。

DOM树作为应用最广泛的HTML网页表达形式，非常适合用来表征HTML网页，因此，本模块将采用DOM树的表达方式。

本模块涉及一下三个功能：

1）网页获取和解析：使用requests库根据URL获取网页数据，并使用BeautifulSoup对网页进行解析。

2）预处理和DOM树构建：使用解析后的BeautifulSoup对象构建DOM树，使用treelib库对DOM树进行存储。其中，预处理包含对HTML节点的处理和CSS的处理，主要包含一下流程：

A. TAG节点信息处理：包含替换掉HTML文件中出现的链接（由于链接较长且无明显意义，容易影响后序的hash），合并节点属性等
B. 静态CSS信息处理：CSS在网页中有三种存在形式，其中内嵌和内联样式可以按照节点信息进行处理，外联CSS由于要多次调用requests库获取，可能影响性能，故此处仅作字符串处理。
C. 动态CSS处理（可选）：由于部分网页存在JavaScript动态控制CSS样式，因此需要借助模拟器进行网页模拟，从而获取网页的渲染后的CSS元素。

3）DOM树合并CSS数据：将CSS数据与DOM树进行合并，最终构建完整的网页DOM树表示。

2.2 网页数据向量化模块

网页数据向量化模块包括两部分功能，分别为特征抽取和特征向量化映射。

1）特征抽取

对生成的DOM树节点进行遍历，抽取单个节点的信息和所有相关子结构的信息，将对应的tag和attrs（包含CSS）信息进行拼接，生成特征字符串。为了降低复杂度，此处仅包含单个节点的子节点和兄弟节点信息，包含越多的子树结构信息可以带来更好的特征刻画，但同时也会加大计算复杂度。

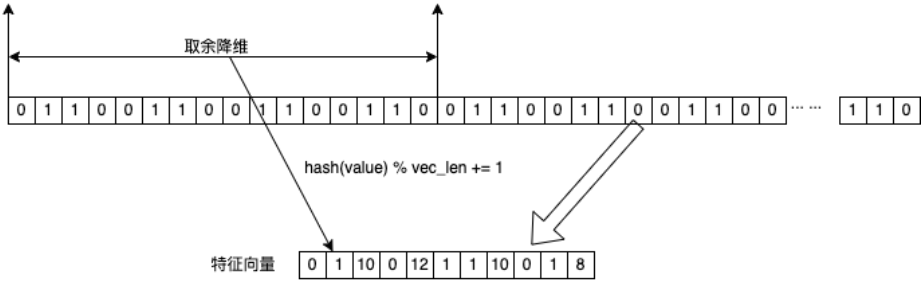
2）特征向量化映射

将每个节点抽取的特征信息字符串进行hash映射达到转换不等长字符串映射等长数值的目的。在此处的系统设计中，每个节点的信息和和其兄弟节点和子节点的信息将被映射成不同的hash值，因此一个节点最多会有三个哈希值作为特征。

需要注意的是，传统的hash函数目的是将原始内容均匀随机映射，因此即便两个字符串只差一个字符，其映射后的hash值也可能差别较大。而基于局部敏感性的hash函数（LSH）在其原理上便解决了这样的问题，相似的序列在经过hash之后，其hash值也比较接近，能比较好的反应序列的相似性。考虑到DOM树的结构和特征均是对应的CSS语句或HTML语句，LSH哈希能较好的反应节点和结构相似的特征，因此本系统选取基于LSH的simhash作为结构序列的特征化映射方式。

由于不同DOM树的节点数并不相同，所以产生的hash值的数量并不相同，无法直接进行比较计算，需要将不同DOM树映射的特征降维为同一长度的向量。考虑到本系统中采用的simhash的特征，即Hamming距离较小的文本较为相似，因此需要设计特殊的方式进行降维映射。

本系统采取下图所示取余降维的方式，将所有的hash值按照预设的dimension进行分段取余，然后将对应的余数作为该hash值此分段特征在最终特征向量上的维度，同时将对应维度的统计值加1。此方法类似于在hash上进行分段近似，从而达到按照simhash特征进行降维的目的，最终特征向量上每个维度上的值都是近似字符串在该维度上的数量。



2.3 相似度计算模块

如2.2中所述，网页数据特征向量化的统计方式其本质是各种不同信息和结构feature数量的降维统计，向量值均为整数，代表一类相似特征的数量。

按照常规关于每个特征维度正交的假设，本文使用Cosine Similarity和Jaccard Similarity来衡量两个DOM树的相似度（值越接近于1越相似，越接近于0越不相似）。

其计算公式如下：

Cosine Similarity:

$$\text{Similarity}\ (\mathrm{A},\ \mathrm{B}) = \frac{\mathrm{A}\cdot\mathrm{B}}{||\mathrm{A}|| \times ||\mathrm{B}||} = \frac{\sum_{\mathrm{i}=1}^{\mathrm{n}}(\mathrm{A_i} \times \mathrm{B_i})}{\sqrt{\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{A_i^2}} \times \sqrt{\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{B_i^2}}}$$

Jaccard Similarity:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

2.4 结果评估模块

如2.3中所述的算法，计算出的相似度是一个数值，需要有一个具体的阈值对比来评估网页是否相似，这本质是一个监督学习的问题，需要一些样本用来构造校验阈值。本章通过设计一部分特殊样本来通过数据分布的方式来决定此阈值，后序可以增加较多的监督样本来对此阈值进行较为准确的更新。

本文采取两种方式构造正向样本，用正向样本中比较小的平均相似度来作为相似度判定的阈值。构造数据的原则为：同一类型外观相似的网站样本和同一网站相同布局的子网页样本。

构建的监督数据集如下：

1）Youtube的视频子网页和Bilibili的视频子网页

2）Baidu和Google的同一关键词的搜索结果子网页

3）Github代码仓首页和Gitee代码仓首页

经过计算，本系统推荐的阈值（大于阈值即可表征为相似）为：

相似度算法	阈值
Cosine Similarity	0.25
Jaccard Similarity	0.20

三、结果评估

本章将评估几个经典例子，用来说明本系统的作用，更多数据应该在增加了监督或半监督数据集以后评估。

同一网站各子网页相似度计算结果如下：

网站名	Cosine Similarity	Jaccard Similarity
Github	0.93	0.71
Google	0.94	0.74
Baidu	0.92	0.84
Bilibili	0.93	0.83
Youtube	0.96	0.81
Gitee	0.86	0.71

同一类型不同网站各子网页相似度计算结果如下：

网站1	网站2	Cosine Similarity	Jaccard Similarity
Gitee	Github	0.57	0.37
Bilibili	Youtube	0.25	0.20
Baidu	Google	0.13	0.04

其中Baidu和Google搜索页面的对比数值偏小，查看原因得知，其布局样式差异较大，针对不同搜索关键词，搜索结果的页面差异也比较大，并非传统认知上的二者相似，其布局如下图所示。

四、大规模场景下如何改进和扩展

在请求量和计算量都较大的大规模场景下，应从资源、架构、流程、结构组件和算法优化等方面考虑，有以下方案可以支撑：

1. 在资源方便，优先基于Spark等的大数据计算系统响应请求，优先动态进行资源扩展。如使用Kafka作为消息中间件，Spark集群用Streaming的方式对每条请求进行响应，计算资源不够时可以动态扩展加载资源。同时可以针对性的使用AI算法对计算资源进行workload调优，以提高计算响应速度或降低成本。
2. 架构上对每个请求中的两个URL的处理应并行处理，每个URL单独进行内容获取、处理和特征向量的生成流程。
3. 流程上，URL获取内容可以通过增加缓存，设置专门的网页获取服务器来进行内容获取，或通过AI算法预测和数据画像的方式提前缓存可能的URL内容，加快网页内容获取速度。
4. 结构组件上，可以使用高性能的树处理库和高性能的hash库来对关键操作进行加速以及高性能的计算库来支持相似度计算。
5. 算法优化上，对DOM树的构建和遍历可以通过充分并行化的方式实现，加快处理速度。对节点特征的抽取和向量化也可以充分进行并行加速。

五、结论与展望

本系统通过对HTML/CSS网页结构相似度计算问题进行系统调研和分析，构建了一套可扩展的高效率相似度计算系统，可以满足问题中的需求，后序可以针对性的对性能和算法准确度等进行继续研究。

5.1 是否有其他方案

可以参考以下方案：

1. 可以考虑按照树的编辑距离计算的方案。
2. 可以考虑将网页存成样式图片，利用图片的SSIM算法进行计算。
3. 可以考虑将DOM树表示为图，利用图的经典算法或Graph embedding等方式来进行特征表征，从而进行计算。
4. 可以使用AI算法在构建监督数据集的基础上，构建分类器进行计算，输入可以是网页结构的DOM树、文本或图片。