PageRank

背景

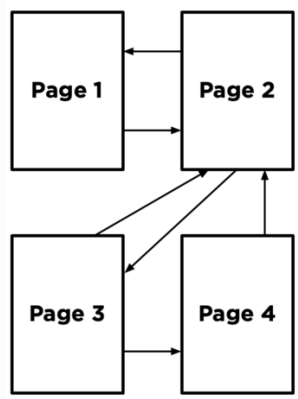
当像谷歌这样的搜索引擎显示搜索结果时，他们是通过在搜索结果中，将更重要和更高质量的网页放在更高的位置，将重要性低的网页放在更靠后的位置。但是，搜索引擎如何知道哪些页面比其他页面更重要？

一种启发式方法是，一个 "重要 "的网页是一个被许多其他网页所链接的网页，可以合理地想象，更多的网站会链接到一个质量较高的网页，而不是质量较低的网页。因此，我们可以设想一个系统，根据每个页面从其他页面传入的链接的数量给它一个等级，而更高的等级将意味着更高的重要性。

在 PageRank 的算法中，如果一个网站被其他重要的网站链接，那么它就比较重要，而来自不太重要的网站的链接的权重就比较低。有多种策略来计算这些排名。

Random Surfer Model

PageRank 的一种方式是使用 random surfer model，该模型考虑了互联网上假设的冲浪者随机点击链接的行为。考虑下面的网页语料库，其中两个页面之间的箭头表示从一个页面到另一个页面的链接。



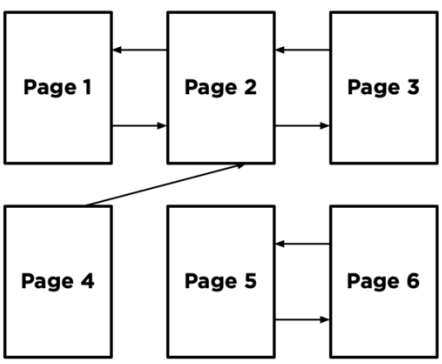
一个用户从一个随机的网页开始，然后随机地选择链接。例如，如果用户在第 2 页，他们会在第 1 页和第 3 页之间随机选择下一步访问（同一页面上的重复链接被视为一个单一的链接，而且从一个页面到其本身的链接也被忽略）。如果他们选择了第 3 页，用户就会在第 2 页和第 4 页之间随机选择下一个访问。

那么，一个页面的 PageRank 可以被描述为一个随机的用户在任何给定时间都在该页面的概率。毕竟，如果有更多的链接到一个特定的页面，那么一个随机的用户就更有可能最终进入该页面。此外，一个更重要的网站的链接比一个不太重要的网站的链接更有可能被点击，而后者链接的页面更少，所以这个模型也可以处理按重要性加权的链接。

对这个模型的一种解释是马尔可夫链，其中每个页面代表一个状态，每个页面有一个过渡模型，在其链接中随机选择。在每个时间步骤，状态切换到当前状态所链接的页面之一。

通过从马尔可夫链中随机抽取状态，我们可以得到每个页面的 PageRank 估计值。我们可以从随机选择一个页面开始，然后不断地随机选择链接，记录我们访问每个页面的次数。在我们收集了所有的样本之后，我们在每个页面上的时间比例可能就是对该页面排名的估计。

但是，只能根据网页上的链接跳转到下一网页，会出现一些问题，参考下面的网页连接图



如果最开始从 page5 出发，那么只能在 page6、page5 之间反复，访问其他的网页的可能性为 0。

为了确保总是能够到达网页语料库中的其他地方，我们将在模型中引入一个阻尼系数 d。在概率为 d 的情况下（d 通常设置为 0.85 左右），从当前页面的链接中随机选择一个。但是在概率为 1-d 的情况下，会从语料库中的所有页面中随机选择一个（包括用户当前所在的页面）。

用户现在开始随机选择一个页面，然后，对于接下来的选择，以 d 的概率随机选择当前页面的一个链接，并以 1-d 的概率随机选择任何页面。记录每个页面被选中的次数，我们可以把在一个给定页面上的状态比例视为该页面的 PageRank。

Iterative Algorithm

也可以用一个递归的数学表达式来定义一个页面的 PageRank。让 PR(p)成为一个给定页面 p 的 PageRank：一个用户出现在页面 p 的概率。

有两种方式可以让一个随机的冲浪者最终出现在该页面上：

（1）在概率为 1-d 的情况下，用户随机选择了一个页面，最后出现在 p 页。

（2）在概率为 d 的情况下，用户跟随一个链接从 i 页到 p 页。

第一个条件在数学上是相当直接的：它是 1-d 除以 N，其中 N 是整个语料库的页面总数。对于第二个条件，需要考虑链接到 p 页的每一个可能的页面 i。对于每一个传入的页面，让 NumLinks(i)为 i 页上的链接数。每一个链接到 p 页的 i 页都有自己的 PageRank，PR(i)代表任何特定时间在 i 页上的概率。由于从第 i 页前往该页的任何一个链接的概率相同，我们用 PR(i)除以链接数 NumLinks(i)，得到在第 i 页上并选择链接到 p 页的概率。

这里给出 PR(p)的公式：

$$PR(p) = \frac{1-d}{N} + d\sum_{i}\frac{PR(i)}{NumLinks(i)}$$

通过迭代的方法计算每个页面的 PageRank：首先假设每个页面的 PR = 1/N，N 为整个语料库的页面总数；然后使用上述公式，基于之前的 PR 值，计算每个页面的新 PR 值；重复上一步，直到所有页面的 PR 值都收敛（只在阈值允许的范围内变动）。

任务

实现 sample_pagerank, iterate_pagerank 两个函数。

（0）transition_model 函数说明

transition_model 的输入为语料库、当前页面、阻尼因子。语料库是一个字典，将一个页面名称映射到该页面所链接的所有页面的集合。页面是一个字符串，代表用户当前所在的

页面。damping_factor（d）是一个浮点数，代表生成概率时要使用的阻尼系数。在概率为 d 的情况下，用户应该以相同的概率选择页面中的一个链接；在概率为 1-d 的情况下，用户以相同的概率选择语料库所有页面中的一个。

该函数返回一个字典，key 是页面，对应 value 为接下来选择该页面的概率。如果一个页面没有外向链接，那么 transition_model 应该返回一个概率分布，在所有页面中以同等概率随机选择。(换句话说，如果一个页面没有链接，可以假装它与语料库中的所有页面都有链接，包括它自己)。

1. transition_model 的输入：
2. 语料库：
   {“1.html”: {“2.html”, “3.html”}, “2.html”: {“3.html”}, “3.html”: {“2.html”}}
3. 当前页面 “1.html”，阻尼系数 0.85
4.
5. transition_model 的输出：{"1.html": 0.05, "2.html": 0.475, "3.html": 0.475}.

（1）实现 sample_pagerank

sample_pagerank 的输入为语料库、阻尼因子、采样总数。该函数返回一个字典，key 是页面，对应 value 为该页面的 PR 估计值（所有 value 的和应该为 1）。

初始随机选择一个页面，作为第一个样本，之后使用 transition_model 获得当前页面去往其他页面的概率字典，通过选择，生成下一个样本。

（2）实现 iterate_pagerank

iterate_pagerank 的输入为语料库、阻尼因子。该函数返回一个字典，key 是页面，对应 value 为该页面的 PR 值（所有 value 的和应该为 1）。

初始设定每个页面的 PR 值为 1/N,N 是语料库中的页面总数,之后每一步使用 iterative algorithm 进行迭代，直到 PR 值收敛，这里设定为 old PR 值与 new PR 值差的绝对值不超过 0.001 即为收敛。

一个没有外向链接的页面被解释为对语料库中的每个页面(包括它自己)都有一个链接。

除了三个需要实现的函数外，请不要修改 pagerank.py 中的任何原有代码，如果有需要，可自定义新的函数以及导入库。更多细节请参考：
https://cs50.harvard.edu/ai/2020/projects/2/pagerank/#random-surfer-model