Hub & authority

覃雄派

Hub & authority

提纲

- HITS算法
- HITS算法的应用
- HITS算法的不足
- HITS算法算法实践

Hub & authority

- HITS(Hyperlink Induced Topic Search)算法
  - 是康奈尔大学的Jon Kleinberg 博士于1997年提出的
  - 这个算法是网页链接分析中非常重要的算法之一

Hub & authority

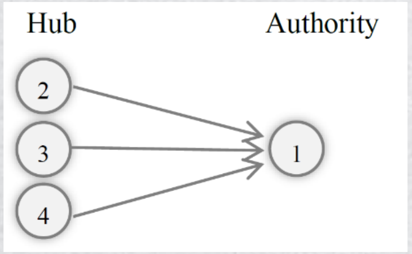
- HITS(Hyperlink Induced Topic Search)算法
  - Authority页面(权威页面)和Hub页面(枢纽页面)是HITS算法最基本的两个概念
    - 所谓Authority页面，是指与某个领域或者某个话题相关的高质量网页
      - 比如，在搜索引擎领域，谷歌、百度的首页就是该领域的高质量网页
      - 而在视频领域，YouTube、优酷、土豆等视频网站的首页，就是该领域的高质量网页
    - 而“Hub”页面，指的是包含了很多指向高质量“Authority”页面链接的网页

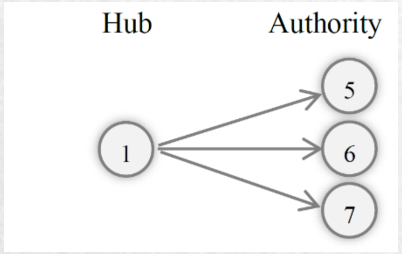
Hub & authority

- HITS(Hyperlink Induced Topic Search)算法
  - HITS算法的目的，是通过一个迭代计算过程，在海量的网页中，找到和用户查询相关的高质量的“Authority”页面和“Hub”页面
  - 其中，“Authority”页面，包含能够满足用户查询的高质量内容
  - 搜索引擎以这些网页为搜索结果，返回给用户

Hub & authority

- HITS(Hyperlink Induced Topic Search)算法
  - HITS算法的基本思想是，“Authority”页面和“Hub”页面具有相互增强的关系
    - (1) 一个好的“Authority”页面会被很多好的“Hub”页面指向
    - (2) 一个好的“Hub”页面会指向很多好的“Authority”页面



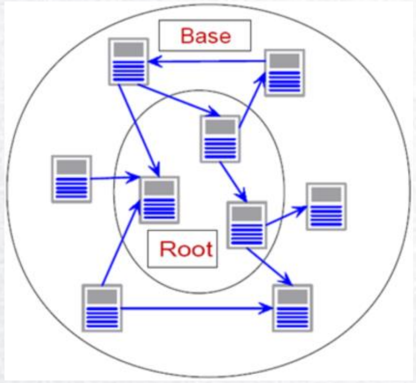


Hub & authority

- HITS(Hyperlink Induced Topic Search)算法
  - 算法的执行过程包括三个大的步骤，分别是构造根集合、扩展根集合，以及迭代计算各个网页的Authority和Hub得分
- **(1) 构造根集合(Root)**
  - 将查询Q提交给基于关键字查询的检索系统，从返回结果页面集合中，取前n(比如n=200)个网页，作为根集合(Root Set)，记为Root
  - 由此可见，Root中的网页数量较少，这些网页是和查询Q相关的网页，Root中包含较多的权威(Authority)网页
  - 这些网页，以及网页之间的链接关系，构成了一个有向图G(V,E)

Hub & authority

- HITS(Hyperlink Induced Topic Search)算法
  - 算法的执行过程包括三个大的步骤，分别是构造根集合、扩展根集合，以及迭代计算各个网页的Authority和Hub得分
- **(2) 扩展根集合(Base)**
  - 在根集合Root的基础上，对网页集合进行扩充，构造集合Base
  - 扩充过程描述如下：凡是与根集内网页有直接链接指向关系的网页都被扩充到集合Base
    - 也就是，无论是有链接指向根集内页面，还是根集页面有链接指向的页面，都被扩充到扩展网页集合Base里
  - HITS算法将在这个扩展网页集合内寻找好的"Hub"页面与好的"Authority"页面



Hub & authority

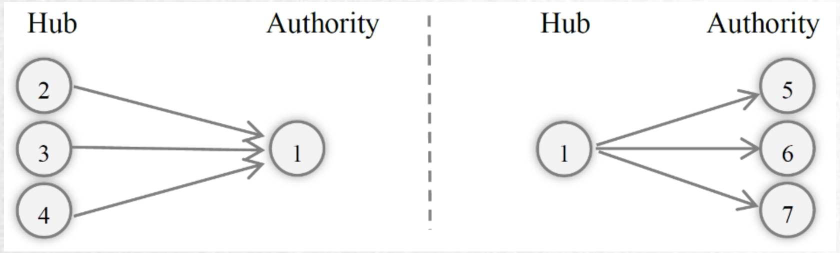
- HITS(Hyperlink Induced Topic Search)算法
  - 算法的执行过程包括三个大的步骤，分别是构造根集合、扩展根集合，以及迭代计算各个网页的Authority和Hub得分
- **(3) 计算扩展集合(Base)中所有页面的Hub得分(枢纽度)和Authority得分(权威度)**

```
$a_0$=1,$h_0$=1 //a, h为向量，各个元素初始化为1，分别对应一个网页的a得分和h得分
t = 1 //迭代时间步为1
do
{
  For each v in V
  {
    $a_t$ (v) = $\sum_{<w,v>\in E} h_{t-1}(w)$
    $h_t$ (v) = $\sum_{<v,w>\in E} a_{t-1}(w)$
    $a_t$ = $a_t$ / | $a_t$ |
    $h_t$ = $h_t$ / | $h_t$|
    t = t +1
  }
} while ( | $a_t$ - $a_{t-1}$ | +| $h_t$ - $h_{t-1}$ | >ε)
Return ($a_t$, $h_t$)
```

Hub & authority

- HITS(Hyperlink Induced Topic Search)算法
  - 算法的执行过程包括三个大的步骤，分别是构造根集合、扩展根集合，以及迭代计算各个网页的Authority和Hub得分
- **(3) 计算扩展集合(Base)中所有页面的Hub得分(枢纽度)和Authority得分(权威度)**

```
$a_0$=1,$h_0$=1 //a, h为向量，各个元素初始化为1，分别对应一个网页的a得分和h得分
t = 1 //迭代时间步为1
do
{
 For each v in V
 {
  $a_t(v) = \sum_{<w,v>\in E} h_{t-1}(w)$
  $h_t(v) = \sum_{<v,w>\in E} a_{t-1}(w)$
  $a_t = a_t / | a_t |$
  $h_t = h_t / | h_t |$
  t = t +1
 }
} while ( $| a_t - a_{t-1} | + | h_t - h_{t-1} | > \varepsilon$ )
Return ($a_t$, $h_t$)
```

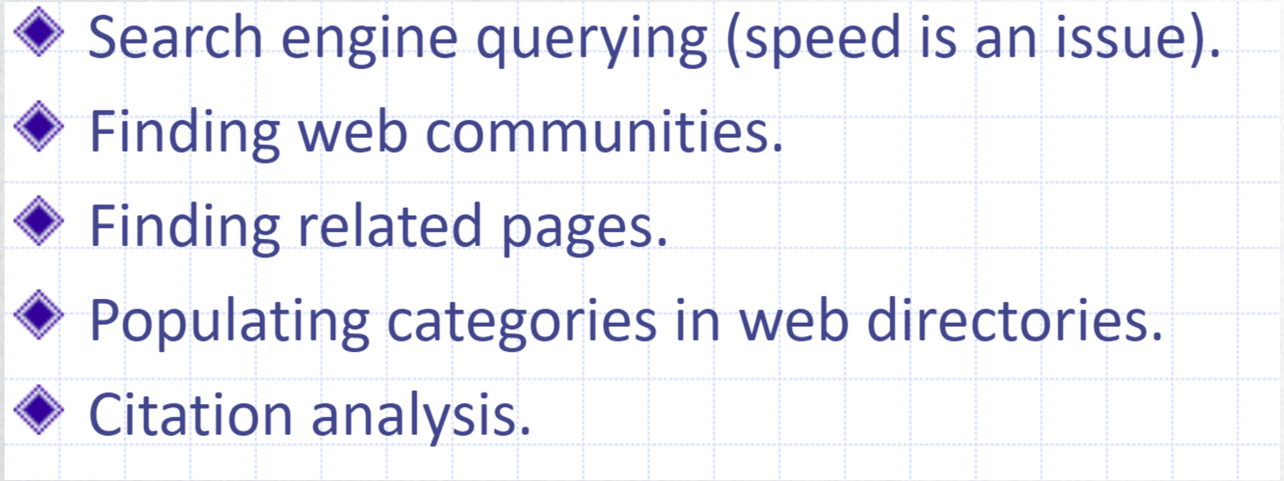


Hub得分和Authority得分的更新方式

Hub & authority

Hub & authority

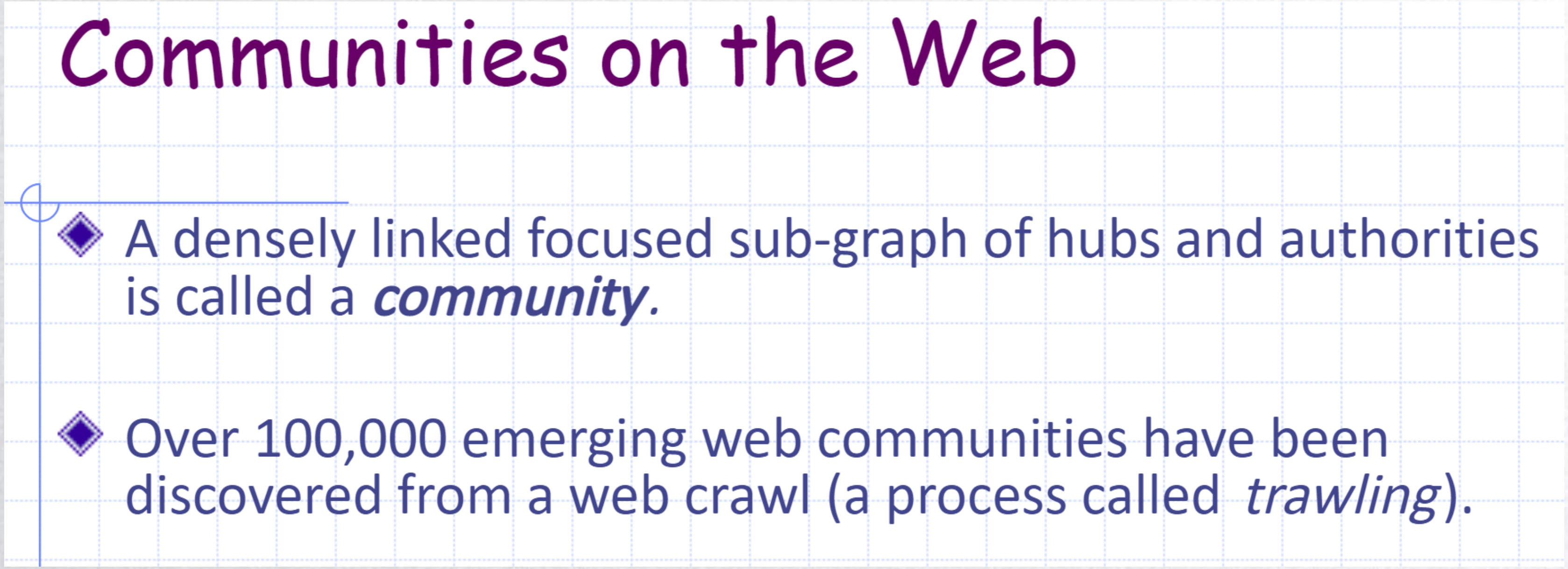
- HITS算法不仅应用在搜索引擎领域
  - 在其他领域也得到了借鉴和应用，并取得了很好的效果



- Search engine querying (speed is an issue).
- Finding web communities.
- Finding related pages.
- Populating categories in web directories.
- Citation analysis.

Hub & authority

- HITS算法不仅应用在搜索引擎领域
  - 在其他领域也得到了借鉴和应用，并取得了很好的效果



Communities on the Web

- A densely linked focused sub-graph of hubs and authorities is called a ***community***.
- Over 100,000 emerging web communities have been discovered from a web crawl (a process called *trawling*).

Hub & authority

Hub & authority

- HITS算法的不足
- HITS算法也有一些不足，这需要我们在实际应用中加以注意。这些不足包括：
  - (1) 结构不稳定：当在已有的“扩充网页集合”内，添加或者删除个别网页，或者改变少数几个链接关系，那么HITS算法的排名结果，就会发生较大的改变

Hub & authority

- HITS算法的不足
- HITS算法也有一些不足，这需要我们在实际应用中加以注意。这些不足包括：
  - (2) 计算效率不高：首先HITS算法必须在接收到用户查询后实时进行计算，此外该算法需要进行多轮迭代计算，才能获得最终结果，其计算效率不高

Hub & authority

- HITS算法的不足
- HITS算法也有一些不足，这需要我们在实际应用中加以注意。这些不足包括：
  - (3) 主题漂移问题：如果在“扩展网页集合”里包含部分与查询主题无关的页面，而且这些页面之间有较多的相互链接指向，HITS算法很可能会给予这些无关网页很高的排名，导致搜索结果发生主题漂移；这种现象，称为“紧密链接社区现象”(Tightly Knit Community Effect)

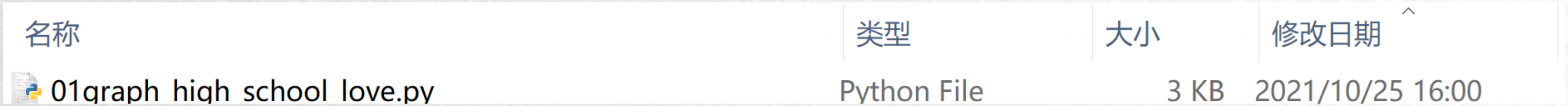
Hub & authority

- HITS算法的不足
- HITS算法也有一些不足，这需要我们在实际应用中加以注意。这些不足包括：
  - (4) 容易被作弊者操纵排名结果：如作弊者可以建立一个网页，页面内容增加很多指向高质量网页或者著名网站的网址，这就是一个很好的Hub页面，之后作弊者在这个Hub页面制作网页链接指向作弊网页，搭个便车，于是可以提升作弊网页的Authority得分

Hub & authority

图数据入门、中心度

- 中心度计算Python实例分析

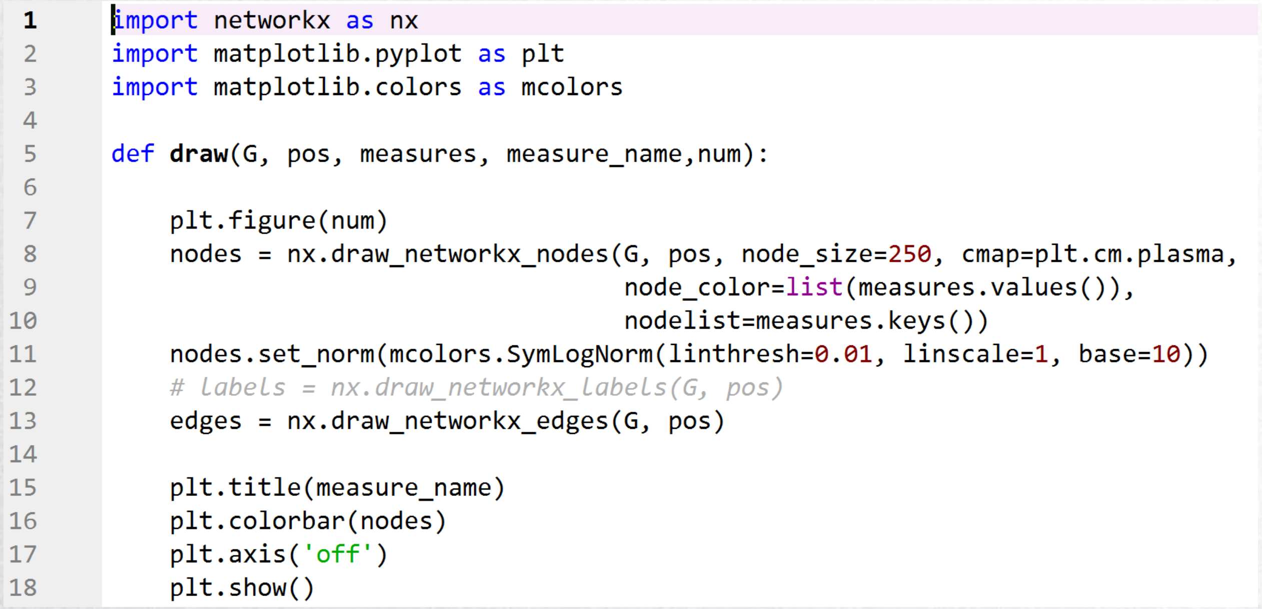


名称	类型	大小	修改日期
01graph_high_school_love.py	Python File	3 KB	2021/10/25 16:00

https://aksakalli.github.io/2017/07/17/network-centrality-measures-and-their-visualization.html

图数据入门、中心度

- 中心度计算Python实例分析
  - 绘制函数



```python
import networkx as nx
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors

def draw(G, pos, measures, measure_name,num):

    plt.figure(num)
    nodes = nx.draw_networkx_nodes(G, pos, node_size=250, cmap=plt.cm.plasma,
                                   node_color=list(measures.values()),
                                   nodelist=measures.keys())
    nodes.set_norm(mcolors.SymLogNorm(linthresh=0.01, linscale=1, base=10))
    # labels = nx.draw_networkx_labels(G, pos)
    edges = nx.draw_networkx_edges(G, pos)

    plt.title(measure_name)
    plt.colorbar(nodes)
    plt.axis('off')
    plt.show()
```

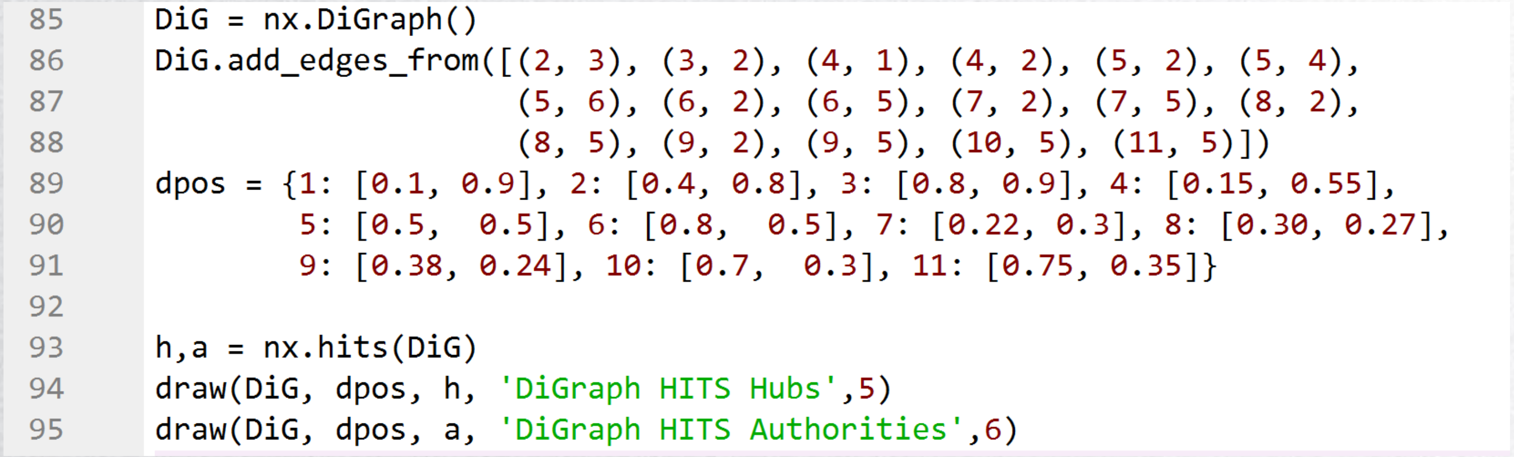
图数据入门、中心度

- 中心度计算Python实例分析
  - 图的创建函数

```python
def school_dating_graph():
    students = set(range(11))
    G = nx.Graph()
    G.name = "Simple Dating Graph"
    G.add_nodes_from(students)
    dating_rel = [(0,3), (1,3), (2,3), (3,4), (4,5), (4,9),
                   (5,6), (6,7), (6,8), (6,9), (9,10)]
    G.add_edges_from(dating_rel)
    # You may want to try automatic layout
    #pos = nx.spring_layout();
    pos = {0: [0.1, 0.6], 1: [0.1, 0.5], 2: [0.1, 0.4], 3: [0.2, 0.5],
           4: [0.3, 0.5], 5: [0.45, 0.7], 6: [0.6, 0.5], 7: [0.7, 0.6],
           8: [0.7, 0.4], 9: [0.45, 0.3], 10: [0.45, 0.2]}
    return G, pos
```

图数据入门、中心度

- 中心度计算Python实例分析
  - 计算Hub & Authority分值

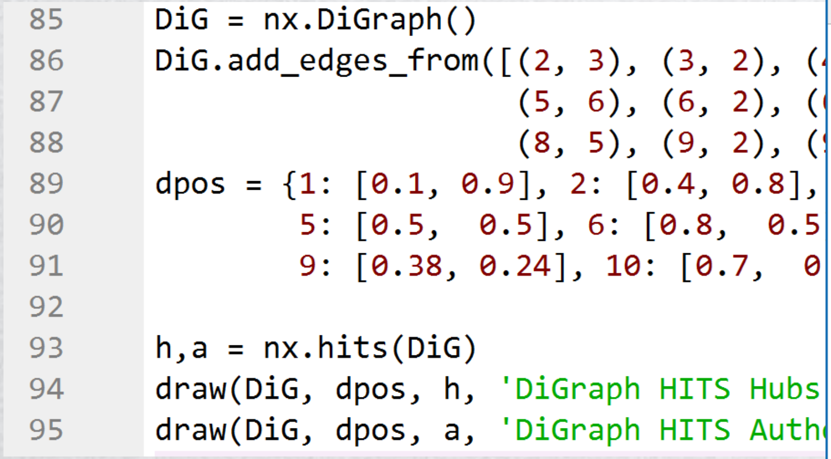


```python
DiG = nx.DiGraph()
DiG.add_edges_from([(2, 3), (3, 2), (4, 1), (4, 2), (5, 2), (5, 4),
                    (5, 6), (6, 2), (6, 5), (7, 2), (7, 5), (8, 2),
                    (8, 5), (9, 2), (9, 5), (10, 5), (11, 5)])
dpos = {1: [0.1, 0.9], 2: [0.4, 0.8], 3: [0.8, 0.9], 4: [0.15, 0.55],
        5: [0.5,  0.5], 6: [0.8,  0.5], 7: [0.22, 0.3], 8: [0.30, 0.27],
        9: [0.38, 0.24], 10: [0.7,  0.3], 11: [0.75, 0.35]}

h,a = nx.hits(DiG)
draw(DiG, dpos, h, 'DiGraph HITS Hubs',5)
draw(DiG, dpos, a, 'DiGraph HITS Authorities',6)
```

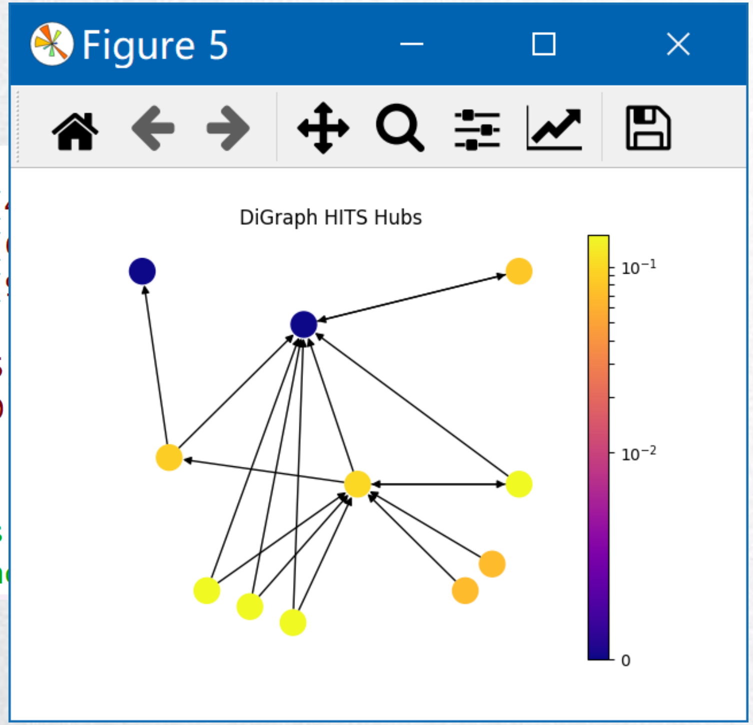
图数据入门、中心度

- 中心度计算Python实例分析
  - 显示Hub & Authority分值



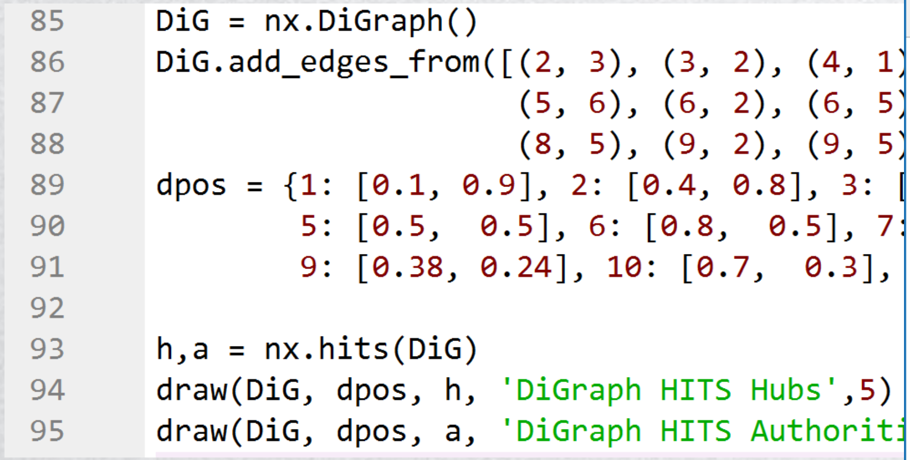
```
DiG = nx.DiGraph()
DiG.add_edges_from([(2, 3), (3, 2), (
                    (5, 6), (6, 2), (
                    (8, 5), (9, 2), (
dpos = {1: [0.1, 0.9], 2: [0.4, 0.8],
        5: [0.5,  0.5], 6: [0.8,  0.5
        9: [0.38, 0.24], 10: [0.7,  0

h,a = nx.hits(DiG)
draw(DiG, dpos, h, 'DiGraph HITS Hubs
draw(DiG, dpos, a, 'DiGraph HITS Auth
```



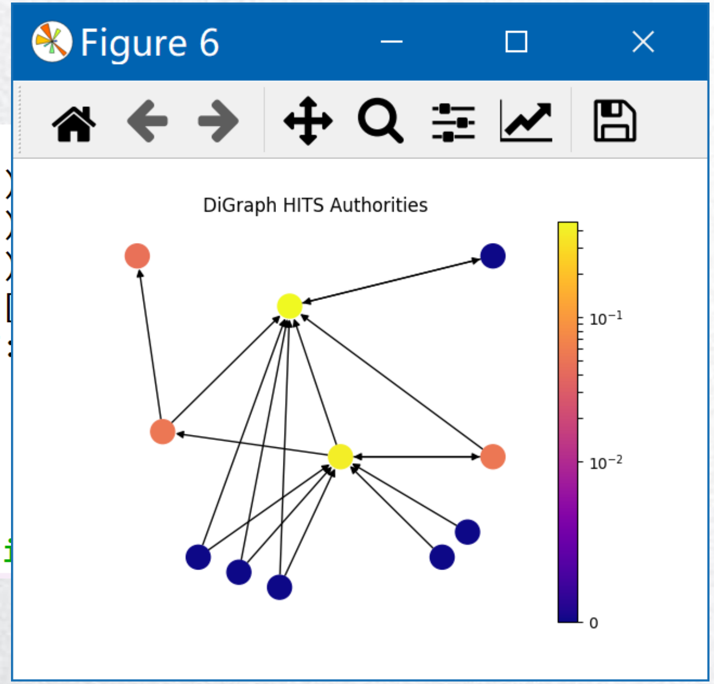
图数据入门、中心度

- 中心度计算Python实例分析
  - 显示Hub & Authority分值



```
DiG = nx.DiGraph()
DiG.add_edges_from([(2, 3), (3, 2), (4, 1)
                    (5, 6), (6, 2), (6, 5)
                    (8, 5), (9, 2), (9, 5)
dpos = {1: [0.1, 0.9], 2: [0.4, 0.8], 3: [
        5: [0.5,  0.5], 6: [0.8,  0.5], 7:
        9: [0.38, 0.24], 10: [0.7,  0.3],

h,a = nx.hits(DiG)
draw(DiG, dpos, h, 'DiGraph HITS Hubs',5)
draw(DiG, dpos, a, 'DiGraph HITS Authoriti
```



Hub & authority

Hub & authority

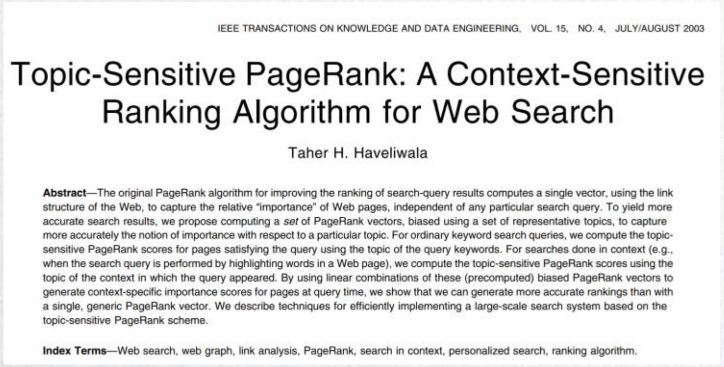
- Relation between HITS, PageRank and LSI
  - HITS algorithm
    - = running SVD on the hyperlink relation (source, target)
  - LSI algorithm
    - = running SVD on the relation (term, document)
  - PageRank on root set R
    - gives same ranking as the ranking of hubs as given by HITS

Mining the Web. Chakrabarti and Ramakrishnan

Hub & authority

- PageRank vs HITS
  - PageRank advantage over HITS
    - Query-time cost is low
      - HITS: computes an eigenvector for every query
    - Less susceptible to localized link-spam
  - HITS advantage over PageRank
    - HITS ranking is sensitive to query
    - HITS has notion of hubs and authorities

- Topic-sensitive Page Ranking [Haveliwala 2003]
- Attempt to make Page Ranking query sensitive



IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 15, NO. 4, JULY/AUGUST 2003

Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search

Taher H. Haveliwala

Abstract—The original PageRank algorithm for improving the ranking of search-query results computes a single vector, using the link structure of the Web, to capture the relative "importance" of Web pages, independent of any particular search query. To yield more accurate search results, we propose computing a *set* of PageRank vectors, biased using a set of representative topics, to capture more accurately the notion of importance with respect to a particular topic. For ordinary keyword search queries, we compute the topic-sensitive PageRank scores for pages satisfying the query using the topic of the query keywords. For searches done in context (e.g., when the search query is performed by highlighting words in a Web page), we compute the topic-sensitive PageRank scores using the topic of the context in which the query appeared. By using linear combinations of these (precomputed) biased PageRank vectors to generate context-specific importance scores for pages at query time, we show that we can generate more accurate rankings than with a single, generic PageRank vector. We describe techniques for efficiently implementing a large-scale search system based on the topic-sensitive PageRank scheme.

Index Terms—Web search, web graph, link analysis, PageRank, search in context, personalized search, ranking algorithm.