Website Tag Propagation

June 25, 2013

Amulya K
amulya.k@iiitb.org

N Puneeth
puneeth.n@iiitb.org

Sindhu Priyadarshini
sindhu.priyadarshini@iiitb.org

International Institute of Information Technology, Bangalore

Advanced Algorithms Project Report

Abstract

A website tag propagation system aims to extend the user given annotations/tags from a manually tagged webpage to other webpages in the webgraph. This problem is particularly relevant to data scientists who have to deal with humongous amounts of webpages and each of the webpages cannot be tagged manually. We are proposing a system where in starting from only ONE initial seed website and tag(s) associated with it we propagate the tag(s) to the relevant websites. We are using two metrics in our system, one is the link analysis through the random surfer model and the other is the document similarity metric.

Project URL: https://github.com/puneethn/WebsiteTagPropagation

Acknowledgement

We would like to extend our deepest regards to our advisor,
Prof. G Srinivasaraghavan for his motivating words and useful suggestions throughout this work. It was a pleasure for us to work under his guidance.
We are very grateful to all the people who have helped and supported us during the project. We would also like to extend our gratitude to all the authors whose papers we have referred during the course of our project.

Contents

List of Figures

1 Introduction

Objective

The main objective of the project is to develop a website Tag propagation system. The system has been implemented by using the Link analysis and Page Level metrics.

Motivation

Ubiquity of the web has given rise to immense rise in the rate of generation of the content. Documents shared by users in such applications vary largely, from scientific publications on CiteU-Like (*http://www.citeulike.org*) to images on Flickr (*http://www.flickr.com*) or bookmarks on del.icio.us (*http://del.icio.us*) or the social networking sites such as Facebook, Twitter, Linked In etc. One standard practice in all these scenarios is to rely on user-provided metadata to foster search capabilities.
Tags are short textual annotations used to describe documents and represent such user-generated metadata in Web 2.0 applications. Tags are essential in resolving user queries targeting shared documents, yet they require human attention to be generated. Users typically tag a small fraction of the shared documents only, leaving most of the other documents with incomplete metadata. This lack of metadata seriously impairs search, as documents without proper annotations are typically much harder to retrieve than correctly annotated documents. Automatic tag propagation techniques can produce good results for specific tag categories; they fail however in general, as automating the labelling of arbitrary documents is an inherently challenging research problem.
In this project we investigate, theoretically and experimentally, the application of Link Analysis to ranking on the Web.

2 Overview

This project is about assigning a descriptor tag (metadata) to any given website. This kind of metadata helps describe an item and allows it to be found again by browsing or searching. Tags are generally chosen informally and personally by the items creator or by its viewer. Starting from one initial seed website and the associated tags the system propagates these tags to other websites which are present in the webgraph which we have got by crawling through the hyperlinks present in my initial webpage.

3 Algorithm

There are two parts to our algorithm.

- Firstly, we calculate the Tag Propagation scores by using the random surfer model and the document similarity metric.
- Next based on this score we simulate the user behavior on our webgraph to arrive at the appropriate websites to which tag(s) could be propagated.

3.1 Random Surfer Model:

Random surfer model to the webgraph which we generate to simulate the user behavior.

- Assume that the web is a graph.
- Surfers randomly click on links, where the probability of picking an outlink from page A is 1/m, where m is the number of outlinks from A.

- Using the theory of markov chain, it can be shown that if the surfer follows links for long enough, the rank of that web page is the probability with which he will visit that page.

Algorithm:

Let us have a web graph represented as graph G with N nodes, (1.....N).
We have an Adjacency Matrix: A
Transition Probability Matrix: P
Let α (usually equal to 0.15), be the probability of teleport operations.
If a row of A has no 1s, set each element to 1/N.
For each of the other rows:

- Divide each 1 in the row by the number of 1's in the row, i.e Number of outgoing links in each row O_i.
- Multiply the resulting row by (1-α)
- Add α/N to every entry of the resulting matrix.

The each entry of the Probability matrix $P_{i,j}$ can be represented as :

$$Pij = \frac{Xij(1-\alpha)}{\sum\limits_{j} Xij} + \frac{\alpha}{N}$$

Only for rows which had at least one non-zero value in the initial adjacency matrix.

Where $X_{i,j}$ is an indicator variable which checks for the existence of edge between two nodes i and j.

$$X_{i,j} = \begin{cases} 1 & \text{if there exists a link between nodes } i,j \\ 0 & \text{otherwise} \end{cases}$$

Correctness of Probability Matrix:

To Show:

$$\sum_j Pij = 1$$

i.e each row sums to one.

Proof:

$$\begin{aligned}
&\sum_j \left(\frac{\alpha Xij}{\sum_j Xij} + \frac{(1-\alpha)}{N} \right) = 1 \\
&= 1 - \alpha + \alpha \sum_j \frac{Xij}{\sum_j Xij} \qquad Let \ \sum_j Xij = k \\
&= 1 - \alpha + \alpha \sum_j \frac{Xij}{k} \\
&= 1 - \alpha + \alpha * \tfrac{1}{k} * k \\
&= 1 - \alpha + \alpha \\
&= 1
\end{aligned}$$

Properties of Markov chain

Recurrence:

Recurrence is ensured by setting the expectation of each element in the probability matrix, to a non- zero value.

$$\begin{aligned}
&E[Pij] = E\left[\frac{(1-\alpha)}{\sum_j Xij} Xij + \frac{\alpha}{N} \right] \\
&= E\left[\frac{(1-\alpha)}{\sum_j Xij} Xij \right] + E\left[\frac{\alpha}{N} \right] \\
&= \frac{(1-\alpha)}{Oi} E\left[Xij \right] + \frac{\alpha}{N} \\
&= \frac{(1-\alpha)}{Oi} \frac{1}{Oi} + \frac{\alpha}{N} \\
&= \frac{(1-\alpha)}{Oi^2} + \frac{\alpha}{N}
\end{aligned}$$

Irreducibility:

α/N term in the expression $P_{i,j}$ ensures irreducibility as each term in probability matrix is non zero.

Aperiodic:

The non zero diagonal element, i.e. $P_{i,i}$ term is non zero implies that the markov chain is aperiodic.

Since, the markov chain satisfies these three properties, there exists a stationary distribution.

$$\exists N,\ \mu_t = \mu_{t+1} \qquad \forall t > N$$

These three properties of irreducibility, recurrence and aperiodicity makes the markov chain ergodic.

Example:

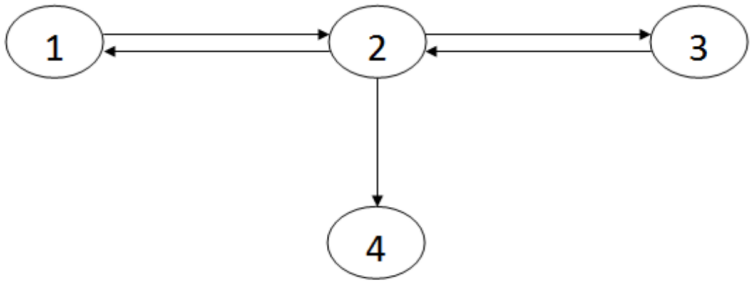

Figure 1: A sample webgraph

We consider an example where we have four webpages/nodes in our system. The incoming and outgoing links between the pages are as shown in the figure.
We get the adjacency matrix of the graph as A, which is further transformed according to our algorithm
α=0.5, N=4.

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & 1 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} \\ 0 & 1 & 0 & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{6} & 0 & \frac{1}{6} & \frac{1}{6} \\ 0 & \frac{1}{2} & 0 & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix} \Rightarrow \begin{pmatrix} \frac{1}{8} & \frac{5}{8} & \frac{1}{8} & \frac{1}{8} \\ \frac{7}{24} & \frac{1}{8} & \frac{7}{24} & \frac{7}{24} \\ \frac{1}{8} & \frac{5}{8} & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

The final matrix gives us the probability matrix for our system which can now be used to obtain the stationary distribution.

3.2 Document Similarity:

Cosine similarity is used to calculate the document similarity score between the documents in question.
Cosine similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them. It gives a useful measure of how similar two documents are likely to be in terms of their subject matter. The cosine of 0 degree is 1, and it is less than 1 for any other angle. It is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a Cosine similarity of 1, two vectors at 90 degrees have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0,1].

$$cos(\Theta) = \frac{x * y}{\|x\| * \|y\|}$$

In our system we compare every node in our webgraph with the reference page

for each tag. We have selected the wikipedia page http://en.wikipedia.org/ of the tag(s) which the user associates with the initial seed url as the reference pages. So every tag would have a document similarity vector representing the cosine similarity between the reference page and every webpage in our webgraph.

3.3 Tag Propagation Score:

The combined tag propagation score for each tag is given by multiplying each row of $P_{\mathrm{i,j}}$ with the corresponding document similarity vector for that particular tag.

For a tag k,

from i = 0 to N

$$TagScore = P_{\mathrm{i,j}} * DocSim_k$$

Where i is constant for every node.

The logic behind multiplying the random surfer probability and the document similarity is that the content of the page is vital to check for the relevance of a particular tag with that webpage and the tag relevance metric needed to be independent of our webgraph to be a universal non biased metric for any given input.
With the product of the two metric being used for simulation, any node which lacks in either of the metric is weeded out of our system hence giving us an accurate output.

3.4 Simulation

There are k different tags associated with our initial seed url. For each tag we calculate the tag propagation score for every node in our webgraph. Based on this tag score, we start simulating the random surfer model for every node in the webgraph.

We perform a random walk for a specific number of hops defined by the 'Mixing Time' parameter, after each walk the system ends up at a particular node. This node is then pushed into a vector which keeps count of the nodes on which we end up during simulation. We run the random walk for certain number of samples. After this we obtain a distribution of nodes across the given number of samples. We repeat this process again to obtain another distribution. We then compare the initial and the final distribution to check if our system has attained stationary distribution. This convergence to a stationary distribution signifies that we have reached the steady state of the simulation. If the stationary distribution is not attained we increment the mixing time by a certain factor and run the whole simulation again. After reaching the stationary distribution the tag is propagated to the nodes whose count is greater than a specific threshold.

We repeat this whole process for every tag which has been assigned to our initial webpage.

```
foreach Tag k do
    init MixingTime = 10;
    init NoOfSimulations = 10000;
    init Threshold = Constant * (NoOfSimulations/NoOfNodes)
    Simulate: for i=(1,2) do
        for (j=0 to NoOfSimulations) do
            FinalNode ← Simulate Markov chain for 'MixingTime' number
            of hops
            hits[i] = hits[i] + FinalNode;
        end
    end
    if (similarity(hits[1],hits[2]) >= 90%) then
        output hits[1]
    end
    else
        MixingTime = MixingTime * 2
        goto Simulate
    end
    if (Count[node] in hits[1] > Threshold) then
        output <Node - Tag k>
    end
end
```

Algorithm 1: Algorithm for tag propagation

4 Architecture

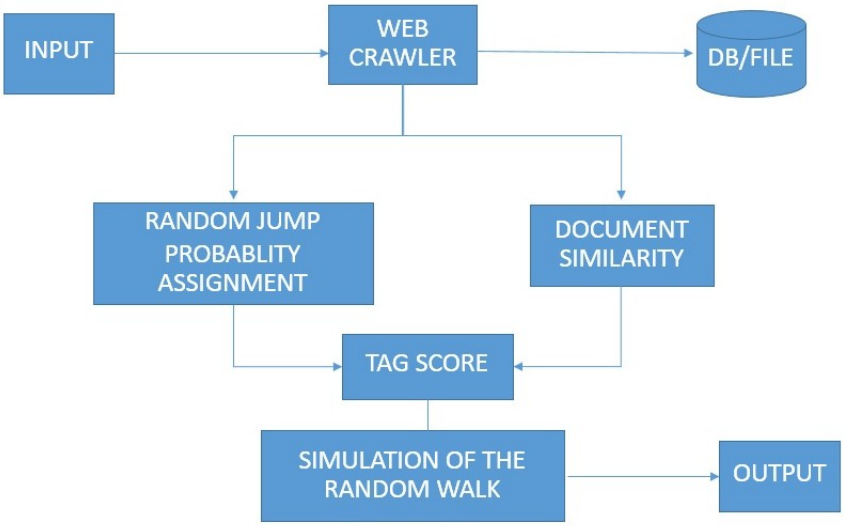

Figure 2: System Architecture

The architecture can be described in a nutshell as follows:

- **Input Module:**
  We obtain the seed url and the associated tag(s) provided by the user. The input can be a wikipedia object or a completely qualified url.

- **Web Crawler:**
  We start from the initial seed url and extract the hyperlinks present in it. We recursively extract the hyperlinks for a specified number hops to form a webgraph.

- **Random Jump Probability module:**
  We apply the random surfer model on our webgraph to obtain the probability matrix $P_{i,j}$ which gives us the probability of jumping from any node i to any other node j in the webgraph.

- **Document Similarity Module:**
  We calculate the cosine similarity between the tag reference page with each webpage node in the webgraph.

- **Tag Propagation score assignment:**
  We combine the probability calculated from the random surfer model and the document similarity score to obtain the tag propagation score for each tag.

- **Simulation of the random walk**
  Based on the tag propagation score we simulate the random walk for a given number of times to accurately propagate the tag to appropriate node.

- **Database/File module:**
  We store the input seed and tag provided by the user. We also store the output <website,tag> pair in the database. We store the webgraph object obtained after web crawling which can be used later to run the simulation.

5 Implementation

5.1 Technology

The project was implemented on Java 7 on eclipse IDE. We use HTTP API to get the content of the page from a specified URL.

We used MYSQL as the backend database to store the seed url and output.

5.2 Parameters

- **Number of samples :**
  The number of simulations of the random walk needs to be significantly

high for the distribution we obtain to be statistically significant.

- **Stationary distribution :**
  The standard distribution similarity value between the two calculated distributions in question was set to be 0.9 for the system to have reached a stationary distribution.

- **Threshold Value :**
  The threshold value for the nodes to be accepted was set at some constant factor of the fraction of the number of simulations and the number of nodes.

- **Mixing Time :**
  Mixing time represents the number of hops needed by a walk to reach the stationary distribution. This was initially set to 10 and was incremented by a factor of two for each failed stationary distribution check.

5.3 Issues

The web being a dynamic entity we faced a few issue during the crawling phase. There were issues of malformed URLs, advertisements, redirected URLs, images, pictures etc. appears in the URLs to name a few. We had to remove such webpages from our webgraph for the system to work.

6 Testing and Performance

Following figures show the snapshot of the results which we manually checked.

	SEED	Tag1	Tag2	Tag3	#Nodes
	http://en.wikipedia.org/wiki/Steven_Gerrard	Liverpool,	Liverpool_F.C.	Football,	608
	Standard dist	0.96463139	0.963260735	0.964149713	
	Mixing time	10	10	10	
	OUTPUT URLs	**Tag1**	**Tag2**	**Tag3**	
1	http://en.wikipedia.org/w/index.php?amp%3Boldid=559367983&title=Steven_Gerrard	Liverpool,	Liverpool_F.C.		
2	http://en.wikipedia.org/wiki/1998-99_FA_Premier_League		Liverpool_F.C.		
3	http://en.wikipedia.org/wiki/1999%E2%80%932000_FA_Premier_League		Liverpool_F.C.		
4	http://en.wikipedia.org/wiki/Defender_(association_football)			Football	
5	http://en.wikipedia.org/wiki/England_national_football_team			Football	
6	http://en.wikipedia.org/wiki/Eric_Cantona		Liverpool_F.C.	Football	
7	http://en.wikipedia.org/wiki/2000%E2%80%9301_FA_Premier_League		Liverpool_F.C.		
8	http://en.wikipedia.org/wiki/2000%E2%80%9301_UEFA_Cup		Liverpool_F.C.		
9	http://en.wikipedia.org/wiki/2001%E2%80%9302_FA_Premier_League		Liverpool_F.C.		
10	http://en.wikipedia.org/wiki/2001_FA_Charity_Shield	Liverpool			

Figure 3: Seed: Steven Gerrard

	Seed URL	Tag1	Tag2	#Nodes
	http://www.espncricinfo.com/	Sport	Cricket	44
	Standard dist	0.996401971	0.998615873	
	Mixing time	10	10	
	OUTPUT URLs	**Tag1**	**Tag2**	
1	http://blogs.espncricinfo.com/surfer/	cricket		
2	http://stats.espncricinfo.com/ci/engine/current/stats/index.html	cricket		
3	http://tv.espn.co.uk	cricket		
4	http://www.espn.co.uk/games/sport/games/games.html?game=cricketconcentration%3BCMP%3	cricket	cricket	
5	http://www.espn.co.uk/games/sport/games/games.html?game=jigsaw%3BCMP%3Dci	cricket	cricket	
6	http://www.espn.com.au/	cricket	cricket	
7	http://www.espncricinfo.com/	cricket		

Figure 4: Seed: Cricinfo

SEED	Tag1	Tag2	No of Nodes
http://www.thehindu.com/	News	Information	3736
standard dist	0.999300539	Empty Set	
Mixing Time	10	10	
URLs	Tag1	Tag2	
Http://www.thehindu.com/		information	
http://www.thehindu.com/features/metroplus/radio-and-tv/article4810690.ece	news		
http://www.thehindu.com/news/cities/chennai/shift-in-news-access-imminent-says-cp-chandrasekhar/article4681301.ece?ref	news		
http://www.thehindu.com/news/resources/the-business-of-news-in-the-age-of-the-internet/article4681152.ece	news		
http://www.thehindu.com/news/resources/the-business-of-news-in-the-age-of-the-internet/article4681152.ece?ref=sliderNews	news		
http://www.thehindu.com/news/resources/the-business-of-news-in-the-age-of-the-internet/article4681152.ece?test=1&textsiz	news		
http://www.thehindu.com/news/resources/the-business-of-news-in-the-age-of-the-internet/article4681152.ece?test=2&textsiz	news		

Figure 5: Seed: The Hindu

SEED	Tag 1	Tag 2	#Nodes	#OP
http://www.johnlennon.com/	Music	Beatles	80	3
Standard Distribution	0.998925754	0.99626345		
Mixing Time	10	10		
URL	Tag1	Tag2		
http://www.johnlennon.com/	music	Beatles		
http://www.johnlennon.com/u	music			
http://www.johnlennon.com/wp-content/themes/jl/favicon.ico	music			

Figure 6: Seed: John Lennon

	Seed	Tag 1	Tag 2	Tag 3	#nodes
	http://en.wikipedia.org/wiki/Uttarahalli	bangalore,	karnataka	india	225
	Standard Dist	0.985868736	0.98589034	1	
	Mixing time	10	10	10	
	OUTPUT URLs	Tag 1	Tag 2		
1	http://en.m.wikipedia.org/wiki/Uttarahalli		karnataka		
2	http://en.wikipedia.org/w/index.php?amp%3Boldid=544444648&title=Uttarahalli	bangalore,	karnataka		
3	http://en.wikipedia.org/wiki/Acharya_Institute_of_Technology		karnataka		
4	http://en.wikipedia.org/wiki/B.M.S._College_of_Engineering		karnataka		
5	http://en.wikipedia.org/wiki/Bangalore	bangalore,	karnataka		
6	http://en.wikipedia.org/wiki/List_of_districts_of_India	karnataka			
7	http://en.wikipedia.org/wiki/Bangalore_-_Mysore_Infrastructure_Corridor	bangalore,	karnataka		
8	http://en.wikipedia.org/wiki/States_and_territories_of_India	karnataka			

Figure 7: Seed: Uttarahalli

Figure 7 shows a part of the results obtained for the input {“http://en.wikipedia.org/wiki/Uttarahalli”, {bangalore, Karnataka} }, where the first part represents the initial seed website and the other part is the three tags that need to be propagated. Tags which have been marked green are those propagated correctly. Yellow represents ambiguity, and red represents wrong propagation.

SEED URL	TOTAL	INCORRECT	AMBIGUOUS	CORRECT	ACCURACY
Steven Gerrard	244	9	18	217	0.88
Cricinfo	10	0	2	8	0.8
The Hindu	7	0	0	7	1
John Lennon	4	0	0	0	1
Uttarahalli	190	4	15	171	0.9

After manually checking for the correctness of the tag propagated, accuracy is calculated as the ratio of correct propagation to the total propagation. The system is found to give an accuracy of about 89.5%. There were 407 correct annotations out of the 455 website-tag pairs which we output.
There were only 13 out of the 455 which were wrongly annotated, which comes up to around 2.8%. Hence, the accuracy of our system is fairly high.

7 Conclusion and Future Work

The link structure of the web is a great source of information. This can be used to pinpoint related websites, even without using any kind of semantic information. Even simple probabilistic graphical models such as the random surfer model can be highly effective in modelling highly dynamic entities such as the web.

In this project, the link structure and document similarity score acted as the two metrics for tag propagation. Google which is considered to be the state of art search engine, uses many more such metrics, estimated to be around 200 and many of those can be used along with these two metrics to further improve the results.

Semantic information and ontologies can be further included to increase the page/document level understanding.

We can further use the Hadoop File system and the associated technologies in that ecosystem for e.g Apache Nutch for web crawling, HBase as the database, Apache Mahout for implementing the Machine Learning algorithms in the future.

8 References

[1] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. In Proceedings of the 7th International World Wide Web Conference, Brisbane, Australia, 1998.

[2] J. Kleinberg. Authoritative sources in a hyperlinked environment. Journal of ACM (JASM), 46, 1999.

[3] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In Research and Development in Information Retrieval, pages 104-111, 1998.

[4] Panayiotos Tsaparas, Link Analysis Ranking, PhD Thesis, Graduate School of Computer Science, University of Toronto, 2004.

[5] P. Tsaparas, Using Non-Linear Dynamical Systems for Web Searching and Ranking , Principles of Database Systems (PODS), Paris, 2004.

[6] WebLA, Java package for webgraph, used under BSD License, Author: Bruno Martins, XLDB Group, Department of informatics, Faculty of sciences, University of Lisbon

[7] Budura, Adriana, et al. "Neighborhood-based tag prediction." The semantic web: research and applications. Springer Berlin Heidelberg, 2009. 608-622.