Link-Prediction in a Hyperlinked Network

Aaron Barlev
University of Maryland
abarlev@umd.edu

May 14, 2019

Abstract

In this paper, we evaluate the feasibility and usefulness of a method to predict the likelihood of the creation of hyperlinks between existing web pages. The application we will examine is in the ranking of search engine results by quality for a given search query.

Search engine algorithms are required to not only collect a large number of relevant web pages to a given search query, but also return those results in an order useful for a user. However, web pages that would be considered useful for one user may not be for another. For this reason, many different approaches can be taken when evaluating the relevance or rank of a given web page returned for a search query. Most search engine algorithms complete this task by evaluating the current quality of a page, usually by examining the incoming and outgoing hyperlinks. With this hyperlink-prediction method, search engine results have the potential to be more forward-looking and evaluate not only the current state of the links between web pages, but also a potential future state with the likelihood of a given hyperlink appearing.

1 Introduction

The linked structure of web pages and the hyperlinks between them is full of information that can be utilized if evaluated in the proper way. This linked structure is equivalent to a graph, where the nodes are represented by distinct web pages and the edges between them are represented by hyperlinks. When evaluated in this way, this data structure is rich in information that can be utilized for various applications. In the context of this paper, the focus will be the relevance of this data structure to locating web pages (nodes) that are likely to be useful for a search engine user.

One method is to evaluate the incoming and outgoing hyperlink edges to rank a given web page on its quality and return the set of web pages in that order. Additionally, the nature of these hyperlinks can be evaluated to determine even more information about a web page. For example, what is the rank of the web page that an incoming edge is coming from? Or, what is the ratio of incoming edges to outgoing edges? However, this information limits the algorithm to the current state of a web page. Some research has been conducted in the area of link-prediction in the context of a social network. By structuring web pages and hyperlinks as a graph, like a social network, parallels can be drawn so that link-prediction can be applied to hyperlinks as well.

2 Prior Work

2.1 The Anatomy of a Large-Scale Hypertextual Web Search Engine

In a 1998 paper out of the Stanford University Computer Science Department called *The Anatomy of a Large-Scale Hypertextual Web Search Engine*, Sergey Brin and Lawrence Page outline the logic and algorithm behind what would become the world's most-used search engine. That paper was called The Anatomy of a Large-Scale Hypertextual Web Search Engine. This search engine and the algorithm behind it was named Google (the common spelling of a very large number) to fit Brin and Page's goal of building massively scaled search engines. In order to index many distinct web pages, a ranking system called "PageRank" was devised [1].

$$PR(A) = (1-d) + d\left(\frac{PR(T1)}{C(T1)} + \cdots + \frac{PR(Tn)}{C(Tn)}\right)$$

Figure 1: PageRank Algorithm

It is assumed that web page A has a set of web pages $T1$ to Tn which link to it. The variable d is a damping factor between 0 and 1. In the initial research paper, Brin and Page state the commonly used value for d was 0.85. $PR(Tn)$ is the PageRank of web page Tn and $C(Tn)$ is the total number of links leaving Tn. The PageRank of any given web page can be calculated using an iterative algorithm. The PageRank of a page essentially represents the probability that an internet user visits that page [1].

PageRank was designed to model user behavior. The probability that a random user visits a given web page is equivalent to that page's PageRank. PageRank was novel not in its ability to retrieve web pages that match a certain search query, but in its approach of ordering search results for users. A search engine is nearly useless for a user if their desired web page appears in the 5th page of results. By ranking a web page using the PageRank of other web pages which have links that point to it, Brin and Page were more successfully able to return satisfactory results for users [1].

The data structures which PageRank works alongside are nearly as important as the algorithm itself. Some unique data structures outlined in this paper by Page and Brin are BigFiles, the repository, and the document index. BigFiles are virtual files which encompass several file systems. The files are addressable by 64 bit integers and handle some specific search engine tasks like allocation and deallocation of file descriptors. The repository is the body of the search engine structure. It stores the entire HTML of all the indexed web pages. It is important to note that a compressed version of each file is being stored, not the raw text. Page and Brin also outline the thought process when selected a compression technique. For their requirements, a technique called zlib was chosen. This compression format provided the optimal values for speed and compression ratio (these two metrics typically have an inverse relationship). The document index has a self-explanatory name. It records information about each document and is easily accessed to support document searching. Some of the information stored includes "current document status, a pointer into the repository, a document checksum, and various statistics" [1].

Other data structures outlined for the purpose of the search engine are the lexicon, hit lists, the forward index, and the inverted index. The lexicon, in its simplest form, is simply the word list of a search engine. Brin and Page describe a lexicon supported by the list of words (each separated by a null character) and a hash table of pointers. Hit lists are lists of occurrences for a certain word in any given document. Position, font, and capitalization data is also stored. The forward

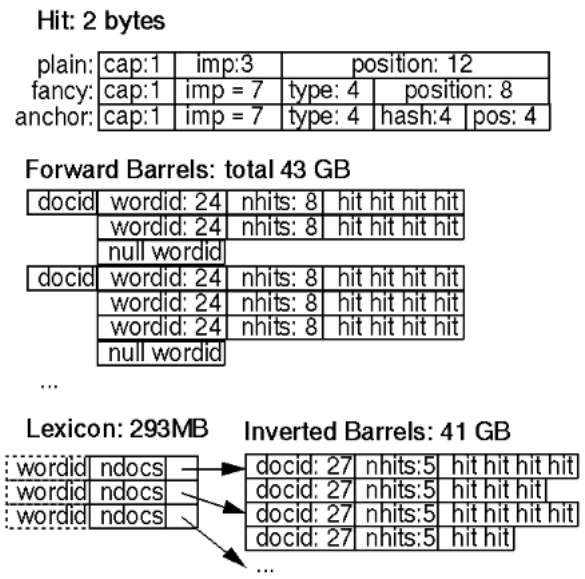

Figure 2: Forward/Reverse Indexes and the Lexicon [1]

index consists of certain wordID's designated to align with a given "barrel". So, when a document contains words that fall into a certain barrel, that is where the docID will be recorded. Finally, the inverted index contains the same barrels as the forward index, except they are completely sorted. The lexicon contains the pointers to the proper barrels for every wordID [1].

Taking a closer look at the data structures which allow for the seamless operation of a search engine provides a deeper understanding of the mechanisms which power such a complex tool. In this way, it becomes possible to construct the most optimal link-prediction algorithms for this specific application.

2.2 Authoritative Sources in a Hyperlinked Environment

Another metric used to return satisfactory results for search engine users was the distinction of a hub and an authority between web pages.

Jon Kleinberg has based *Authoritative Sources in a Hyperlinked Environment* on the notion that the hyperlink network on the World Wide Web (www) is an enormous source of information that can be utilized with proper analysis. When an internet user utilizes an engine to search the www, a list of results is returned that are relevant to the user's query. However, in this paper, Kleinberg is focusing on the keyword of relevance. The degree of relevance that the resulting pages represent helps to define the quality of the results of a search engine. However, relevance is mostly a subjective value that requires human evaluation to determine. Nonetheless, he posits that a search engine which takes a longer period to return a result of greater value to the user than current options would be useful [2].

Any collection V of hyperlinked pages can be illustrated as a directed graph $G =< V, E >$. The web pages are represented by the nodes V and the links between two pages are represented by directed edges (p, q), an element of E. The out-degree of a node p is the number of nodes it is linked to and the in-degree is the number of nodes linked to it. Within G exist two types of valuable web pages: hubs and authorities. An authority is the primary source on a given topic. For example, the University of Maryland website *www.umd.edu* is an authority for a search query of "University

of Maryland". A hub is not a primary source but contains hyperlinks to many authorities for a certain topic. A high-quality hub is a page that points to many high-quality authorities and a high-quality authority is a page that points to many high-quality hubs [2].

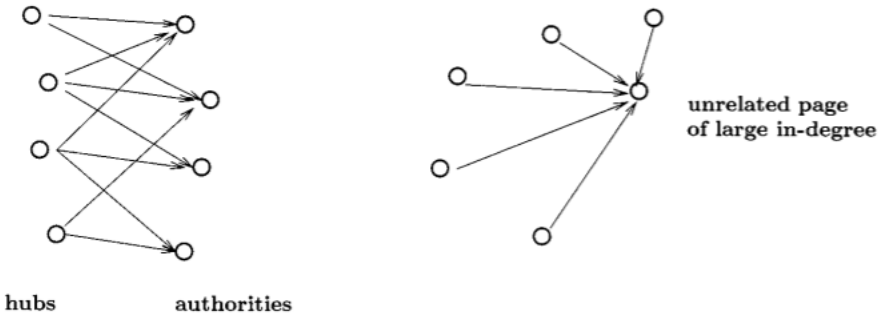

Figure 3: Hubs vs. Authorities [2]

The in-degree ranking is highly valuable when differentiating between hubs and authorities. Typically, authorities will have high in-degrees from a variety of hubs (Figure 3). The algorithm used to differentiate must account for the occasional unrelated page with a large in-degree from a variety of other web pages. These hubs and authorities are called a "mutually reinforcing relationship" by Kleinberg. Typically, high-quality authorities are pointed to by high-quality hubs, and the high-quality hubs are the ones that point to high-quality authorities [2].

$$x^{<p>} \leftarrow \sum_{q:(p,q)\in E} y^{<q>}$$

Figure 4: The J Operation

$$y^{<p>} \leftarrow \sum_{q:(p,q)\in E} x^{<q>}$$

Figure 5: The O Operation

To maintain and update numerical weights for each page p, an iterative algorithm is sued to obtain a non-negative authority weight $x^{<p>}$ and a non-negative hub weight $y^{<p>}$. $x^{<p>}$, the authority weight, equates to the sum of $x^{<q>}$ for all q pointing to p. This is called the J operation by Kleinberg. $y^{<p>}$, the hub weight, equates to the sum of $x^{<q>}$ for all q pointed to by p. This is called the O operation by Kleinberg. These two operations more easily allow hubs and authorities to be distinguished from each other, and therefore higher quality search results to be curated [2].

2.3 The Link Prediction Problem for Social Networks

In *The Link Prediction Problem for Social Networks*, David Liben-Nowell and Jon Kleinberg outline their research into the performance of link-predication metrics. Specifically, in the context of social networks.

A social network is a structure in which the nodes are people/groups (in a social context) and the edges are the "interaction, collaboration, or influence" between them. One of the computational problems that exist regarding social networks is the link-prediction problem. Basically, how can one predict the edges that will be added to existing nodes in a social network over a given period [3].

The social network $G =< V, E >$ contains edges $e =< u, v >$, an element of E, which represent an interaction that took place between u and v at a time t(e). An ideal link-prediction model would be able to provide the probability of an interaction between two given edges $< x, y >$ with a high degree of accuracy. Various predictors exist for this task; some examples are the graph distance, common neighbors, and Jaccard's coefficient models [3].

Liben-Nowell and Kleinberg were able to test the accuracy of various link-prediction models using real-world data sets. The data used was the co-authorship network of the arXiv. Five distinct sections were used: astro-ph (astrophysics), cond-mat (condensed matter), gr-qc (general relativity and quantum cosmology), hep-ph (high energy physics–phenomenology), and hep-th (high energy physics–theory). Various measures were used to evaluate performance, but the Katz Centrality Algorithm proved to perform consistently well on all data sets. The Adamic/Adar index was another high performer [3].

$$score(x, y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{log \mid \Gamma(z) \mid}$$

Figure 6: Adamic/Adar Algorithm

The Adamic/Adar algorithm essentially computes the degree of relation between two nodes in order to determine the probability of a direct edge appearing between them. Within the algorithm, $\Gamma(v)$ represents the neighbors for any given node v. So, z represents the common neighbors of both nodes (x and y). Thus, to derive the prediction score, the sum of 1 divided by the logarithm of the number of neighbors for each common neighbor z is calculated [3].

$$score(x, y) = \sum_{l=1}^{\infty} \left[\beta^l \cdot \mid paths_{(x,y)}^{<l>} \mid \right]$$

Figure 7: Katz Algorithm

The Katz algorithm also calculates a score representing the strength of the probability that a link will appear between two nodes x and y. Within the algorithm, β is simply a parameter that can be varied. The paths value is the set of all length-l paths from x to y. For the unweighted Katz algorithm, the paths value equals 1 if a collaboration exists and 0 if one does not. For weighted, the paths variable equals the number of times x and y have collaborated on a path of length l. Thus, to compute the score, the sum of β^l multiplied by each paths value (weighted or unweighted) of length l from 1 to infinity is computed [3].

3 Novel Work

3.1 Search Engine Applications of Link-Prediction

Insights gained from the research conducted by Brin, Page, Liben-Nowell, and Kleinberg can allow for other algorithms to determine and affirm the results users are seeking when searching the web. Even slight search engine optimization algorithms can save internet users a significant amount of time. To a certain degree, optimal search results for one user may not be optimal for another. This discrepancy also varies according to search terms, as well. However, drawing from the previously outlined research into link-prediction in social networks, algorithms designed to optimize the search engine results for users seeking to locate links between web pages that do not yet exist can be derived.

Essentially, Liben-Nowell and Kleinberg's research on link-prediction in social networks can be applied to networks of web pages. Hyperlinks that are highly likely to be created between two web pages (but do not yet exist) can be accounted for when returning search results to users. The predictors that resulted in the highest performance metrics would be the most optimal choice for this application. That would likely be the Katz measure and its variants. The Adamic/Adar index also performed well in the task of predicting future links [3].

Link prediction can be used to provide search query results that will likely be relevant soon. In order to use a link-prediction algorithm with the context of search query results, it is necessary to first define a graph. Given a list of web pages which match a provided search query, a network (graph) could be structured using the hyperlinks to create directed edges between pages. A hyperlink on a web page indicates a directed edge going from that web page to wherever the hyperlink leads. However, in the context of this link-predicting algorithm, the graph being evaluated would only contain nodes representing web pages that matched the initial search query. This would save a significant amount of computing power when compared to running the link-prediction algorithm before first narrowing the search to relevant web pages.

Typically, a ranking system would be used to return the search query matches in an order that would be most useful to a user. PageRank is one method of organizing results [1]. It is also common to balance the prevalence of hubs and authorities in search results [2]. However, this more forward-looking search engine algorithm would prove useful for users seeking to make connections between websites and topics that do not yet exist. This sort of technology could have applications in areas where novel information is regarded highly. For example, journalism or academia. A journalist seeking to draw connections between topics that have yet to be uncovered would have significant use for a search engine algorithm of this kind. The algorithm may be able to predict connections between different forums or organizations that have yet to appear. Link-predication in the context of the web pages would also have far-reaching relevance in academia. A PhD student seeking to embark on novel research may utilize a link-predicting search engine to examine the relevance of a new research area due to its predicted connection to a known research area. In addition to specific applications of such an algorithm, link-prediction can enhance the quality of standard search engine results. The algorithms behind popular internet search engines could potentially also account for high strength predicted links when calculating a web page's rank to allow for more forward-looking search results.

3.2 Implemented Algorithm

Taking the lessons learned from link-prediction in a social network, the Katz algorithm can be modified in several ways to work effectively for hyperlink-prediction in the context of a search

$$strength(a,b) = \alpha \cdot C(a)_{<1>} + \sum_{l=1}^{N} \left[\beta^l \cdot C(a,b)_{<l>} \right]$$

Figure 8: Hyperlink-Prediction Algorithm

engine. The modified algorithm computes the strength (confidence) that a new hyperlink will be created in the future from a given web page a to a given web page b.

Inspired by the PageRank algorithm [1], $C(a)_{<1>}$ represents the number of links leaving web page a. Simply put, the number of outgoing edges from a. This value multiplied by α represents the "adjacency strength". In the context of PageRank, having many outgoing edges would lower the score. However, in the context of link-prediction, the opposite is true. In theory, having many existing hyperlinks on a page increases the probability that a new hyperlink will be created to a given outside web page. As outlined previously, these types of pages with many outgoing edges have been defined as hubs by Kleinberg [2]. Hubs should be favored as a root web page in this algorithm due to the greater likelihood that a new outgoing edge will be created in the future. For example, if a web page currently contains 100 hyperlinks and that number is increased by 5 percent over the next 6 months, 5 hyperlinks will be added. However, if a web page containing 20 hyperlinks increased at the same rate, only 1 new hyperlink would be created.

The latter portion of this algorithm closely resembles the Katz centrality algorithm. This value is more representative of the final strength value. $C(a,b)_{<l>}$ represents the number of paths from a to b of exactly length l. The sum of this number multiplied by a parameter β raised to the power of l, for all l from 1 to N, is taken. N represents the total number of web pages indexed in the graph. With the possibility of the existence of cycles in the graph, paths could theoretically continue endlessly. Thus, the graph is only scoured for path of a maximum length N, equal to the number of indexed web pages. This sum represents the "path strength". The two parameters α and β can be adjusted to optimize the algorithm for an application. The algorithm was designed such that $0 < \alpha < \beta < 1$.

3.3 Pseudo-code

Algorithm 1 Hyperlink-Prediction Strength

```
1: G: a graph of web pages (nodes) and hyperlinks (directed edges)
2: a, b: root, target
3: N: total number of nodes
4: α, β: optimization parameters
5: procedure STRENGTH(G, a, b, N, α, β)
6:     adjacencyStrength ← α * number of outgoing edges from a
7:     pathStrength ← 0
8:     for l from 1 to N do
9:         pathStrength += β^l * number of paths from a to b of length l
10:    end for
11:    return adjacencyStrength + pathStrength
12: end procedure
```

3.4 Java Code

A program was written in Java to test the Hyperlink-Prediction Algorithm. In the program, there is a `HyperlinkGraph` class which defines a list of nodes and a hash map where the key is represented by a node and the value is list of adjacent nodes. For this application, each node is a web page and each adjacent node is connected by a hyperlink. For the program, the input search query graph is an example created with the results returned from a PageRank-like algorithm for a search term of "Maryland". The output of the program is the relative strength that a hyperlink will be created in the future for various pairs of web pages existing in the graph.

The strength value is computed by solving two sub-problems and returning their sum. First, the adjacency strength is calculated. This is simply the length of the adjacency list for a multiplied by α. Next, the path strength is calculated. $C(a,b)_{<l>}$ represents the total number of paths that exist between a and b of exactly length l. This value is determined using a depth-first search of the graph and is then multiplied by β raised to the power of l. l is a variable and ranges from 1 to N, the total number of distinct nodes in the graph. This sum is computed using a for-loop, and the depth-first search for $C(a,b)_{<l>}$ is done recursively. Finally, the path strength and adjacency strength are summed and returned, representing the strength that a hyperlink will be created between two distinct web pages a and b.

3.4.1 Input

Within the `HyperlinkGraph` class, the `hyperlinkPredictionStrength` method was used to test the Hyperlink-Prediction Algorithm. This method requires six input parameters. First, and most importantly, is the input graph. The input graph created for this purpose consists of eleven distinct web pages and a diverse set of edges between them. The input parameters also include the starting and ending web pages for which the strength between them is being calculated. These variables are stored as `String a` and `String b`. Finally, an integer `N` stores the total number of web pages in the graph and two doubles `alpha` and `beta` store the α and β parameters, respectively.

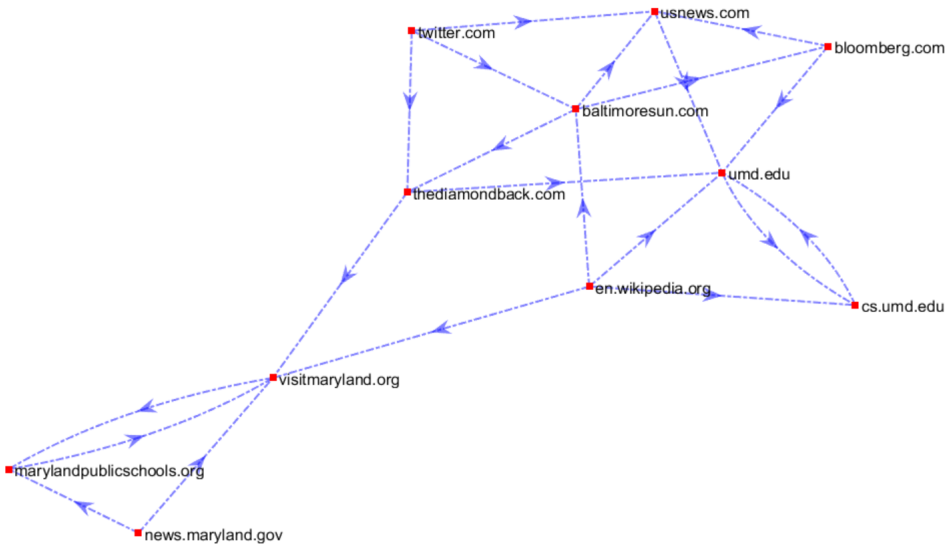

Figure 9: Input Graph

3.4.2 Output

The output of the `hyperlinkPredictionStrength` method is simply a value stored as a double datatype representing the hyperlink-prediction strength between the two web pages initially passed as parameters. This strength value is entirely relative to the strength values of other pairs. Its value depends mostly on α and β. For comparison to other data sets, the values should first be normalized.

3.4.3 Results

A thorough testing of the input graph was completed by varying several different parameters. First, every possible combination of nodes in the graph were tested (excluding pairs of the same node and pairs of nodes with an edge already existing between them). Then, the highest performers were selected from the group. To be considered a highest performer, the strength value must be greater than or equal to the sum of α and β. This testing was conducted for three different α - β options: 0.1 - 0.5, 0.4 - 0.6, and 0.05 - 0.95. The raw (non-normalized) strength values for the highest performing web page pairs are listed in Tables 1, 2, and 3.

From	To	Strength
twitter.com	umd.edu	1.549
baltimoresun.com	umd.edu	1.466
twitter.com	cs.umd.edu	0.924
baltimoresun.com	cs.umd.edu	0.883
en.wikipedia.org	marylandpublicschools.org	0.816
twitter.com	visitmaryland.org	0.8
en.wikipedia.org	usnews.com	0.775
bloomberg.com	cs.umd.edu	0.7
en.wikipedia.org	bloomberg.com	0.65
baltimoresun.com	visitmaryland.org	0.633

Table 1: Test 1, highest prediction strength pairs for $\alpha = 0.1$, $\beta = 0.5$

For Test 1, the α and β values were set to 0.1 and 0.5, respectively. This would be considered a fairly conservative test. The *pathStrength* was weighted five times less heavily than the *adjacencyStrength*, and both parameters were set to a value of 0.5 or less. This test can be considered a baseline for what default parameters should be set to.

The *adjacencyStrength* was weighted more aggressively in Test 2. The *adjacencyStrength* was considered two-thirds as strongly as the *pathStrength*. The α value was set to 0.4 and β to 0.6. The algorithm is designed such that the *adjacencyStrength* should never exceed the weighting of the *pathStrength*. So, α can never exceed β. Note that the strength values achieved in this test are higher than those of the first test due to the higher parameter settings.

Test 3 consisted of the most extreme weighting out of the three tests. The α value was set to 0.05 and β to 0.95. This means that the *adjacencyStrength* was nearly not considered at all compared to the *pathStrength*. 95% of each strength value was derived from the *pathStrength*. Due to the unusually high β value, the strength values were also noticeably higher. The maximum strength value for Test 3 was 20.976 compared to 3.524 from Test 2 and 1.549 from Test 1.

From	To	Strength
twitter.com	umd.edu	3.524
baltimoresun.com	umd.edu	3.213
twitter.com	cs.umd.edu	2.588
baltimoresun.com	cs.umd.edu	2.405
en.wikipedia.org	marylandpublicschools.org	2.358
en.wikipedia.org	usnews.com	2.176
twitter.com	visitmaryland.org	2.095
en.wikipedia.org	bloomberg.com	1.96
baltimoresun.com	visitmaryland.org	1.759
twitter.com	marylandpublicschools.org	1.735
bloomberg.com	cs.umd.edu	1.695
en.wikipedia.org	news.maryland.gov	1.6
twitter.com	bloomberg.com	1.56
baltimoresun.com	marylandpublicschools.org	1.535
thediamondback.com	marylandpublicschools.org	1.359
twitter.com	news.maryland.gov	1.2

Table 2: Test 2, highest prediction strength pairs for $\alpha = 0.4$, $\beta = 0.6$

From	To	Strength
twitter.com	umd.edu	20.976
twitter.com	cs.umd.edu	18.314
baltimoresun.com	umd.edu	14.821
baltimoresun.com	cs.umd.edu	13.547
twitter.com	visitmaryland.org	7.393
bloomberg.com	cs.umd.edu	7.343
en.wikipedia.org	marylandpublicschools.org	6.726
twitter.com	marylandpublicschools.org	6.49
baltimoresun.com	visitmaryland.org	3.864
thediamondback.com	marylandpublicschools.org	3.814
baltimoresun.com	marylandpublicschools.org	3.679
en.wikipedia.org	usnews.com	1.96
en.wikipedia.org	bloomberg.com	1.103
twitter.com	bloomberg.com	1.052

Table 3: Test 3, highest prediction strength pairs for $\alpha = 0.05$, $\beta = 0.95$

4 Conclusions

The results produced through testing the Hyperlink-Prediction Algorithm provided a significant amount of valuable insight into the performance and potential of the algorithm. For each of the α - β pairs tested, the results showed noticeable differences but with important consistencies. Within the algorithm, a larger α value increases the importance of the *adjacencyStrength*. Essentially, the number of outgoing edges of a node will be weighted more heavily. A larger β value increases the weight of the *pathStrength* (related to the number of possible paths between two nodes). Therefore:

Test 1 would be considered moderate, Test 2 places an emphasis on the *adjacencyStrength*, and Test 3 favors pairs with a higher *pathStrength*. These variations between the tests are certainly clear in the results. For example, *en.wikipedia.org* has the largest number of outgoing edges between all nodes (4). Therefore, a pair starting at *en.wikipedia.org* in Test 2 (where the importance of outgoing edges is strengthened) should have a higher relative strength value than that same pair in Test 3 (where the importance of outgoing edges is lessened). The strength values of each pair in different tests can be compared by dividing the value by the maximum strength recorded in its respective test. Looking at the pair of *en.wikipedia.org* to *marylandpublicschools.org*: a normalized value of 0.669 is recorded for Test 2 and 0.321 for Test 3. Thus, the relative weighting of each portion of the Hyperlink-Prediction Algorithm played a role in which web page pairs were preferred in each test.

The common pairs between all data sets can be considered the ultimate performers, with the highest hyperlink-prediction strength out of all the nodes. For all three data sets, the highest-predicted web page pair is from *twitter.com* to *umd.edu*. By examining the graph in Figure 9, the reasoning is clear. Both nodes share many common neighbors, and the twitter.com node has many outgoing edges. Other pairs common to all three tests are *baltimoresun.com* to *umd.edu*, *twitter.com* to *cs.umd.edu*, *baltimoresun.com* to *cs.umd.edu*, *en.wikipedia.org* to *marylandpublic-schools.org*, *twitter.com* to *visitmaryland.org*, *en.wikipedia.org* to *usnews.com*, *bloomberg.com* to *cs.umd.edu*, *baltimoresun.com* to *visitmaryland.org*, and *en.wikipedia.org* to *bloomberg.com*. The compiled rankings of the top five web page pairs in order of likeliness to form an edge is outlined in Table 4.

Rank	From	To	Average Normalized Strength
1	twitter.com	umd.edu	1.00
2	baltimoresun.com	umd.edu	0.855
3	twitter.com	cs.umd.edu	0.735
4	baltimoresun.com	cs.umd.edu	0.633
5	en.wikipedia.org	marylandpublicschools.org	0.506

Table 4: Web page pairs with the highest normalized prediction strengths

The Hyperlink-Prediction Algorithm and its Java implementation appear to be an effective mechanism for determining the connection strength between any two web pages in a graph. The algorithm can successfully be adjusted for different networks through the variation of the α and β parameters, allowing for the prediction emphasis to be placed on different aspects of a web page. The algorithm returns highly distinct values for each web page pair, regardless of the α and β parameters, allowing for the clear differentiation of hyperlink-prediction strengths between any two web page pairs. The testing conducted proves true potential viability and usefulness for this algorithm when predicting future connections between related web pages in a hyperlink graph.

References

[1] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine, 1998.

[2] Jon Kleinberg. Authoritative sources in a hyperlinked environment, 1999.

[3] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks, 2007.