MIS Quarterly

RESEARCH ARTICLE

RECOMMENDATION NETWORKS AND THE LONG TAIL OF ELECTRONIC COMMERCE

Gal Oestreicher-Singer
Recanati Graduate School of Business, Tel Aviv University, Tel Aviv 69978 ISRAEL {galos@post.tau.ac.il}

Arun Sundararajan
Stern School of Business, New York University, 44 West 4th Street,
New York, NY 10012 U.S.A. {asundara@stern.nyu.edu}

Appendix A

Algorithm for Data Collection

We use two computer programs for data collection. The first collects graph information and the second collects SalesRank information. Both use Amazon.com's XML data service. This service is part of the Amazon Web Services, which provide developers with direct access to Amazon's platform and databases.

Graph Collection

The program (crawler) which collects the graph starts at a popular book. It then traverses the co-purchase network using a depth-first search. Intuitively, in a depth-first search, one starts at the root (in our case, the one popular book chosen) and traverses the graph as far as possible along each branch before backtracking. At each page, the crawler gathers and records information for the book whose webpage it is on, as well as the co-purchase links on that page. The ASINs of the co-purchase links are entered into a LIFO (Last-In-First-Out) stack. If the algorithm finds it is on the page of a product that it has visited already, it backtracks and returns to the most recent product it has not yet finished exploring. The program terminates when the entire connected component of the graph is collected.

For example, in the graph in Figure A1, the nodes are numbered in the order in which the crawler will traverse the graph. In this case, the collection starts at node 1. Its co-purchase links are nodes 2, 6, and 7. Therefore, those numbers are added to a LIFO stack. The script will then proceed to node 2, whose co-purchases are nodes 3, 4, 5, and thus, those numbers will be added to the LIFO stack, which will now include: 3, 4, 5, 6, and 7. The script will continue to node 3. Since there are no co-purchase links to that node, it will move on to node 4. In the same way, the script will collect data about node 5, node 6, and node 7.

Since node 7 has co-purchase links—nodes 8 and 9—they will be added to the stack. After visiting nodes 8, 9, and 10, the data collection will terminate. As can be seen, the script only stops once it has collected information about the entire connected component. The collection of the entire connected component on Amazon.com takes between four and five hours. The script is run each day at midnight.

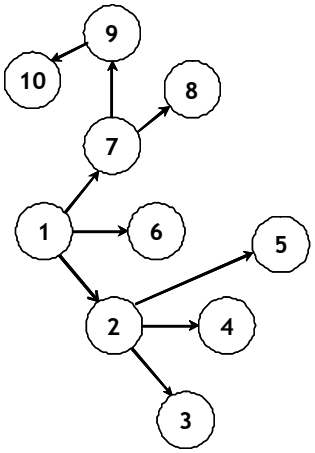

Figure A1. Depth-First Search Used for Graph Traversal

SalesRank Collection

A second computer program collects the demand information for all books on the graph every 3 hours for the 24-hour period following the collection of the graph. This script collects the SalesRank of each book that has ever appeared on the graph. Therefore, it follows the sales of some books that are no longer on the graph.

Appendix B

Converting SalesRank to Demand

SalesRank is a number associated with each product on Amazon.com that measures the product's demand relative to that of the other products sold on Amazon.com. The lower the number is, the higher the sales of that particular product. The SalesRank of a book is updated each hour to reflect recent and historical sales of every item sold on Amazon.com.

A formula to convert SalesRank information into demand information was first introduced by Goolsbee and Chevalier (2003). Their goal was to estimate demand elasticity. Their approach was based on making an assumption about the probability distribution of book sales, and then fitting some demand data to this distribution. They chose the standard distributional assumption for this type of rank data, which is the Pareto distribution (i.e., a power law). In the Pareto distribution, the probability that an observation's value exceeds some level S is an exponential function

$$\Pr(s > S) = \left(\frac{k}{s}\right)^{\theta} \tag{22}$$

where k and θ are the parameters of the distribution. The more important parameter is θ, the shape parameter that indicates the relative frequency of large observations. If θ is 2, for example, the probability of an observation decreases in the square of the size of the observation. With a value of 1, it decreases linearly.

For a given book, the number of books that have sales greater than those of that book is just one less than the book's rank. Therefore, the fraction of all books that have sales greater than those of a particular book is just [*SalesRank* – 1]/*TotalNumberOfBooks*. If there is a sufficient number of books to eliminate the approximation introduced by discreteness, then one can replace the equation above with:

$$\frac{[SalesRank - 1]}{TotalNumberOfBooks} = \left(\frac{k}{Demand(j)}\right)^{\theta} \tag{23}$$

Taking logs on both sides, and substituting θ with $-1/b$, this translates ranks into sales as follows:

$$Log[Demand(j)] = \alpha + bLog[SalesRank(j)] \tag{24}$$

The parameters a and b were estimated by Goolsbee and Chevalier using several parallel methods: by using data from the *Wall Street Journal* book sales index, which gives the actual quantity sold; by using sales information given by a publisher who sells on Amazon.com; and by conducting an experiment, buying copies of books with a steady SalesRank.

In a later study, Brynjolfsson et al. (2003), used data provided by a publisher selling on Amazon.com to conduct a more robust estimation of the parameters of the formula. They estimate the parameters as: $a = 10.526$, $b = -0.871$.

Appendix C

A More Detailed Description of PageRank

Let u be a web page. Let $F(u)$ be the set of pages u points to, and let $B(u)$ be the set of pages that point to u. Let $N(u)=|F(u)|$ be the number of links from u, and let c be a factor used for normalization (so that the sum of rank across all web pages is constant). A simple ranking, $R(u)$, is defined as

$$R(u) = c \sum_{v \in B(u)} \frac{R(v)}{N(v)} \tag{25}$$

This is a simplified version of PageRank. The rank of a page is divided among its forward links evenly to contribute to the ranks of the pages to which they point. Note that $c < 1$ because there are a number of pages with no forward links, and their weight is lost from the system. The equation is recursive, but it may be computed by starting with any set of ranks (commonly, equal rank for all pages) and iterating until convergence.

Stated another way, let A be a square matrix with the rows and columns corresponding to numbered web pages. Let $A(u,v) = \frac{1}{n(u)}$ if there is an edge from u to v, and $A(u, v) = 0$ otherwise. If we treat the rankings as a vector R over the linked pages, we have

$$R = cAR \tag{26}$$

So R is an eigenvector of A with eigenvalue c. In fact, the interesting one is the dominant eigenvector of A. It may be computed by repeatedly applying A to any non-degenerate start vector.

There is a small problem with this simplified ranking function. Consider two webpages that point to each other but not to any other page. Suppose there is some webpage that points to one of them. Then, during iteration, this loop will accumulate rank but never distribute any rank (since there are no outgoing edges). The loop forms a sort of trap which is called a *rank sink*. To overcome this problem of rank sinks, the damping factor $(1 - \alpha)$ is introduced. The normalization factor c is then set to α. Thus, the full ranking formula is

$$R'(u) = \alpha \sum_{v \in B(u)} \frac{R'(v)}{N(v)} + (1 - \alpha) \tag{27}$$

For further details and extensions, see Langville and Meyer (2005).

Appendix D

Correlation Matrix

The correlation matrix for our variables is presented in Table D1.

Table D1. Covariance Matrix

	RevenueGini	*AvgPagRank*	*PageRankVar*	*PagreRankKurtois*	*AvgDemand*	*Size*	*ListPrice*	*SalePrice*	*Mixing*	*Clustering*
RevenueGini	1.00									
AvgPageRank	0.10	1.00								
PageRankVar0.17	0.17	0.58	1.00							
PageRankKurtosis	0.09	-0.56	-0.25	1.00						
AvgDemand	0.55	0.05	0.06	0.10	1.00					
Size	0.19	0.19	0.15	0.18	0.03	1.00				
ListPrice	-0.13	0.22	0.16	-0.30	-0.12	-0.06	1.00			
SalePrice	-0.14	0.20	0.17	-0.28	-0.12	-0.06	0.94	1.00		
Mixing	-0.10	0.13	-0.09	0.10	-0.05	0.36	0.00	-0.02	1.00	
Clustering	-0.10	0.08	-0.23	0.17	0.14	-0.05	-0.41	-0.43	0.23	1.00

References

Brynjolfsson, E., Hu, Y. J., and Smith, M. D. 2003. "Consumer Surplus in the Digital Economy: Estimating the Value of Increased Product Variety at Online Booksellers," *Management Science* (49:11), pp. 1580-1596.

Goolsbee, A., and Chevalier, J. A. 2003. "Measuring Prices and Price Competition Online: Amazon and Barnes and Noble," *Quantitative Marketing and Economics* (1), pp. 203-22.

Langville, A., and Meyer, C. 2005. "Deeper Inside PageRank," *Internet Mathematics* (1:3), pp. 335-380.