MS&E 235: Lecture 11
Prof. Ashish Goel
May 7, 2009
Notes by Rio Goodman

Lecture 11: PageRank, Search and Prediction Markets

In the previous two lectures, we learned how to compute PageRank so that we could assign consistent values to the reputations of web pages. In this lecture, we will see how to apply PageRank to search. We will also introduce prediction markets.

Search results

This section details how a hypothetical search engine could use PageRank to speed up search.

Search engines cannot compute PageRank dynamically every time a user runs a search. Instead, they compute PageRank ahead of time, using static link analysis, and store indexed lists of web pages that can then be searched very quickly. A traditional index is a short section in the back of a book that lists all the major words used in the book, and makes it easy to find these words. Here, a search engine will actually use a *reverse index*, which makes it easy to find every web page that contains a given search word. Since pages with higher PageRank will appear earlier in search results, a logical way to store the indexed list is in order of PageRank. When a user searches for "shoe," the search engine will not want go through every web page looking for the word "shoe," and then shuffle the pages around to be in order of decreasing PageRank. Instead, the search engine uses the reverse index to find the pages containing "shoe," and since the index is already in order of PageRank, the pages can be displayed in the same order.

This procedure works for single word searches, but many searches contain multiple words. For example, if a user searches for "bush league," some kind of intersection between the two reverse indexes must be obtained. Lets say the PageRanks in the indexes look like this:

Bush	League
0.8	0.4
0.2	0.2
...	...

These lists will be extremely long, perhaps 10,000,000 results each. When dealing with long lists such as these, it is important to establish consistent ordering. In the lists above, we know that the first page in the "bush" list cannot be the same page as the first page in the "league" list, because it has a different PageRank. We can also conclude that the first page in the "bush" list cannot appear anywhere in the "league" list, because the first page in the league list has PageRank 0.4, so all the other pages in the "league" list must have PageRanks lower than 0.4, and thus cannot have PageRank of 0.8. higher PageRank than the first page in the "league" list. Therefore, we can eliminate the 0.8 page from the set. With this page eliminated, we can now use the same logic to eliminate the 0.4 page. Using this type of algorithm, we can quickly find the top ranked pages. Finding *all* the results, however, will still take a long time.

Notice that for multi-word search, PageRank ordering works well because the ordering is not tied to the search words themselves. If our notion of reputation was dependent on the search word itself, then we would not be able to compare entries in one list to entries in the other, and returning search results would take much longer.

In this example, PageRank not only gives us a reputation for each page, but also makes the search more efficient.

Personalized PageRank

It is possible to alter the PageRank equations to better approximate special circumstances. The original definition of PageRank is the fraction of time a random surfer will spend on the web page, given that the surfer clicks on a link on the current page at random with probability $(1-\epsilon)$ and travels to a new page in the network with probability ϵ. But we can model different behavior by modifying this surfer's algorithm. For example, the random surfer could randomly click on a link on the current page with probability $(1-\epsilon)$ and revert to page v_0 with probability ϵ. We can adapt the equations to fit this new definition of PageRank. The new equations will be:

$$\pi_{v_0} = \epsilon + (1-\epsilon) \sum_{(w,v)\in E} \frac{\pi_w}{d_w}$$

$$\pi_x = 0 + (1-\epsilon) \sum_{(w,v)\in E} \frac{\pi_w}{d_w}$$

$$\text{for } x \neq v_0$$

This could be useful in many situations. For example, v_0 could be the list of a user's bookmarks, or pages which a user deems trustworthy, resulting in personalized PageRank. Or, v_0 could be a list of pages containing a particular word to find word-dependent reputation. Or, v_0 could be a list of pages which are deemed to be definitively non-collusive by a search engine.

Collusion

Web pages can collude to improve their collective PageRank. This is less of an issue with PageRank than with naïve PageRank, but it is still a problem. An example of collusion is a number of hotels that have many links to each other, but very few links to or from the rest of the web. This creates a net for a PageRank monkey. If the monkey arrives at any one of the hotel web pages, it will be stuck in the community of hotels until it randomly jumps again. Thus, the PageRank of all the pages in the community is artificially high. It is rare that the monkey enters this set of web pages, but once it does, it stays there for a long time. Attempting to increase the PageRank of a web page by placing unnecessary links on the page is known as "link spam."

The problem of finding sets of collusive web pages, even approximately, is NP-hard in the worst case (adversarial). Unfortunately, the worst case will often be realized because collusive web pages do not want to be discovered, and thus will place links to deliberately make the community appear legitimate.

Since this problem is NP-hard, search engines (Google/Yahoo!/MSN) implement secret heuristics to identify link spam. The search engines rely on the fact that the heuristics are secret. Web pages who are attempting to create collusive communities must determine what the heuristics are, and then attempt to overcome them. Once the web pages figure out the heuristics that the search engines are using, then the search engines must implement new heuristics, and the cycle repeats. This is similar to an arms race, or the cryptography/deciphering that happened during World War II. These are examples of zero sum games, which means someone may win in the short run, but no one wins in the long run.

So far, there is no robust online reputation system that does not rely on secrecy or on some form of economic incentives/costs, but perhaps there will be in the future.

Prediction markets

A prediction market is a market in which people predict outcomes of events, and receive a payoff if they predict the outcome accurately. Examples of prediction markets include the Iowa Electronic Exchange and Intrade.

The way a simple prediction market would work is there are "yes" shares and "no" shares. When someone buys into the market for \$1, they receive one "yes" share and one "no" share. Users are then allowed to trade their shares on an open market. Finally, when the event is observed, if the outcome is "yes," then users

are paid \$1 for every "yes" share they are holding. If the outcome is "no," then users are paid \$1 for every "no" share they are holding. Notice that if a user buys in, and makes no trades, then he/she is guaranteed to break even.

To further describe the market, we define c_Y as the price of a "yes" share, and c_N as the price of a "no" share. At any time, $c_Y + c_N = 1$, otherwise there would be arbitrage opportunities.

In an example prediction market with N participants, where $p_y(i)$ is the probability estimate of the i^{th} individual, the participants beliefs are ordered from lowest to highest in the following graph.

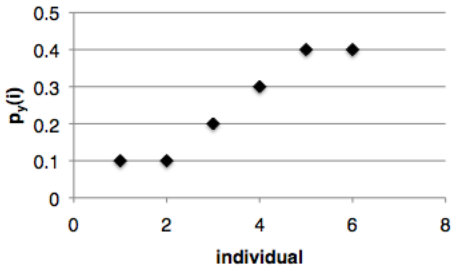

Figure 1: Prediction market

Suppose that in this market, the current prices are $c_Y = 0.35$, $c_N = 0.65$. All the individuals who believe the probability of a yes outcome to be less than 0.35 will buy no shares, and the individuals who believe the probability of a yes outcome to be more than 0.35 will buy yes shares. So in this example, the first four users would buy no shares, and the last two would buy yes shares.

But suppose that each individual is allowed (or wants to hold) only one share. Then the only stable prices in this market are $0.2 \leq c_Y \leq 0.3$, because at these prices, three individuals will buy yes shares, and three will buy no shares. If $c_Y > 0.3$, then there will be a surplus of yes shares, and the price will drop because no one wants them. The stable price, therefore, is the median $p_y(i)$.

If outsiders perceive the current c_Y to be cheap, in other words, if they think that the probability of a yes outcome is higher than the market has it valued at, then they will enter the market.

Prediction markets are believed to be a much more accurate way of predicting outcomes of events than methods such as polling. There are two main reasons for this. First, because if the prices are way off, then a lot of individuals will buy or sell. Second, experts will make fine-grained trades, increasing the accuracy further.

Prediction markets are suited for objective events, i.e., events whose likelihood is not affected by the market itself.