FixingOpicScoring

Fixing the OPIC algorithm in Nutch

Ken Krugler

21 August 2006

DRAFT

Introduction

WARNING - what I've dumped in below still needs to be completed, and cleaned up.

The goal of this proposal is to define the changes needed for Nutch to do a real implementation of the Adaptive On-line Page Importance Calculation (Adaptive OPIC), as described by this paper.

Currently Nutch uses a partial implementation that's mostly useful for crawling "important" pages first. A partial description of this can be found here.

Unfortunately this has a number of problems, specifically:

1. You can't recrawl, at least not without having the recrawled pages get inflated scores. This isn't such a problem when the score is just being used to sort the fetch list, but when the score is also used to determine the page's boost (for the corresponding Lucene document) then this is a Bad Thing. And that's currently what Nutch does.
2. The total score of the "system" as defined by the graph of pages continues to increase, as each newly crawled page adds to the summed score of the system. This then penalizes pages that aren't recrawled, as their score (as a percentage of the total system) keeps dropping.

There were other problems that have recently (as of August 2006) been fixed, such as self-referential links creating positive feedback loops.

The Adaptive OPIC Algorithm

I'm going to try to summarize the implementation proposed by the original paper, as I think it applies to Nutch, but there are still a few open issues that I'm trying to resolve with the authors.

1. Each page has a "current cash value" that represents the weight of inbound links.
2. Whenever a page is processed, the page's current cash is distributed to outlinks, and zeroed.
3. Each page also has a "historical cash value" that represents the cash that's flowed into the page in the last iteration. Initially this starts out as 0.0.
4. The score of a page is represented by the sum of the historical and current cash values.
5. There is one special, virtual root page that has bidirectional links with every other page in the entire web graph.
    a. When a crawl is initially started, the root page has a cash value of 1.0, and this is then distributed (as 1/n) to the n injected pages. When more pages are injected, it's not clear what happens, but I imagine that some of the cash that has accumulated in the root page is distributed to the injected pages, thus keeping the total "energy" of the system constant.
    b. Whenever a page is being processed, the root page can receive some of the page's current cash, due to the implicit link from every page to the root page. So called "dangling nodes", i.e. pages without outlinks, give all of their cash to the root page.
6. To handle recrawling, every page also has the last time it was processed. In addition, there's a fixed "time window" that's used to calculate the historical cash value of a page. For the Xyleme crawler, this was set at 3 months, but it seems to be heavily dependent on the rate of re-crawling (average time between page refetches). *We could use a value derived from fetchInterval.*
7. When a page is being processed, its historical cash value is calculated from the page's current cash value and the previous historical cash value. The historical cash value is estimated via interpolation to come up with an "expected" historical cash value, that is close to what you'd get if every page was re-fetched and processed at the same, regular interval. Details are below.

Details of cash distribution

While distributing the cash of a page to the outlinks, there are a few details that need to be handled:

1. Some amount of the cash goes to the root page, while the rest of the cash goes to the real outlinks. If a page is a leaf (no outlinks) then all of the cash goes to the root page. The ratio of real/root can be adjusted to put greater emphasis on new pages versus recrawling, but the OPIC paper is a bit fuzzy about how to do this properly.
2. Self-referential links should (I think) be ignored. But that's another detail to confirm.
3. There's a mention in the paper to adjusting the amount of cash given to internal (same domain) links versus external links, but no real details. This would be similar to the current Nutch support for providing a different initial score for internal vs. external pages, and the "ignore internal links" flag.
4. I'm not sure how best to efficiently implement the root page such that it efficiently gets cash from every single page that's processed. If you treat it as a special URL, then would that slow down the update to the crawldb?
5. The OPIC paper talks about giving some of the root page cash to pages to adjust the crawl priorities. Unfortunately not much detail was provided. The three approaches mentioned were:
    a. Give cash to unfetched pages, to encourage broadening the crawl.
    b. Give cash to fetched pages, to encourage recrawling.
    c. Give cash to specific pages in a target area (e.g. by domain), for focused crawling.

Details of historical cash calculation

When a page is being processed, its historical cash value is calculated in one of two ways, based on the page's delta time (time between when it was last processeed, and now).

1. If the delta time is >= the time window, then the historical cash value is set to the page's current cash value * (time window/delta time). So using the above, if the page's cash value is 10, and the delta time is 6 months, then the historical cash value gets set to 10 * (6/3) = 5.0.
2. If the delta time is < the time window, then the historical cash value is set to the page's current cash value + (historical cash value * (time window - delta time)/time window). This is kind of odd, but basically it assumes that the "weight" of the past (historical cash value saved for the page) decreases over time, and the current cash will increase as more pages are processed (and thus their inbound contributions contribute to this page's current cash).

There's an issue with new pages, as these will have a current cash value, but no historical cash value. The OPIC paper says that they (Xyleme) use an average value for recently introduced pages, but there aren't any more details. I'm trying to get some clarification.

opic.pdf illustrates some issues with the current OPIC implementation in Nutch. opic.ods contains supporting calculations.