Honours Year Project Report

Predicting Web 2.0 Thread Updates

By

Shawn Tan

Department of Computer Science

School of Computing

National University of Singapore

2012

Honours Year Project Report

Predicting Web 2.0 Thread Updates

By

Shawn Tan

Department of Computer Science

School of Computing

National University of Singapore

2012

Project No: H079830
Advisor: A/P Kan Min-Yen
Deliverables:
Report: 1 Volume

Abstract

In this report, we study a problem and design an efficient algorithm to solve the problem. We implemented the algorithm and evaluated its performance againts previous proposed algorithms that solves the same problem. Our results show that our algorithm runs faster.

Subject Descriptors:
C5 Computer System Implementation
G2.2 Graph Algorithms

Keywords:
Problem, algorithm, implementation

Implementation Software and Hardware:
python, bash

Acknowledgement

List of Figures

List of Tables

Table of Contents

Chapter 1

Introduction

With the advent of Web 2.0, sites with forums, or similar thread-based discussion features are increasingly common. Our goal in this thesis is to create an algorithm that can predict when updates in such threads will occur.

Table **??** shows us that many of the popular Web 2.0 sites have comment features. This suggests that content on the web is increasingly being created by users alongside content providers.

In an increasing number of cases, news travels more quickly through online community discussions than through traditional media. Users also typically discuss purchased products bought online on forums, and companies that want to get timely feedback about their product should turn to data mined from such sites.

Web crawling is largely IO-bound, a large portion of the time spent crawling is waiting for the server to supply a response to the request issued by the crawler. However, for sites with a large number of pages (as in popular forum sites), make this infeasible in practice. On top of the usual requests it has, it then has to deal with repeated requests from such a crawler. Most sites do not mind some additional bandwidth, but if it gets excessive, it may be construed as a Denial-of-Service attack. At best, the site may deny any further requests from the crawler, and at worst the large number of requests may bring down the site.

A simple method to reduce the amount of polling needed is to use the average time differences between previous posts to estimate the arrival of the next one, and to abstain from polling until the estimated time.

	T	FB L	FB S	G +1	L	DL	C	PV	Follows
http://www.lifehacker.com	1	1	1				1	1	
http://digg.com/	1	1			1	1	1	1	
http://9gag.com/	1	1	1	1	1		1	1	
http://www.flickr.com/					1		1	1	
http://news.ycombinator.com/					1		1		
http://stackoverflow.com/					1		1	1	
http://www.youtube.com/					1	1	1	1	
http://www.reddit.com/					1	1	1		
http://www.stumbleupon.com/					1		1	1	
http://delicious.com/	1	1					1	1	1

Table 1.1: Features of popular Web 2.0 sites

T = Twitter mentions

FB L = Facebook Likes

FB S = Facebook Shares

G +1 = Google +1

L = Likes (Local)

DL = Dislikes (Local)

C = Comments

PV = Page Views

Follows = Site-local feature for keeping track of user's activities

A key observation in our work is that the contents of the thread may also influence the discussion and hence the rate of commenting. We believe that the content of the thread has information that can give a better estimate of the time interval between the last post and a new one.

For example, a thread in a technical forum about a Linux distribution may start out as a question. Subsequent questions that attempt to either clarify or expand on the original question may then be posted, resulting in a quick flurry of messages. Eventually, a more technically savvy user of the forum may come up with a solution, and the thread may eventually slow down after a series of messages thanking the problem solver.

Let us define all such thread-based discussion styled sites as forums. Ideally, an incremental crawler of such user-generated content should be able to maintain a fresh and complete database of content of the forum that it is monitoring. However, doing so with the previously mentioned naive method would (1) incur excessive costs when downloading un-updated pages, and (2) raise the possibility of the web master blocking the requester's IP address.

Our high level goal: To come up with a suitable algorithm for revisiting user discussion threads, based on the discussion content in the thread. In this project, we focus on forum threads. We demonstrate three different methods for achieving this using regression methods, and also propose a new metric for measuring the timeliness of such a model that balances between the model's timeliness and bandwidth consumption.

In Chapter 2, we explore the related work dealing with predicting web page updates and metrics to measure the performance of such algorithms. In Chapter 3, we discuss the methods that we have come up with to tackle the problem, while Chapter 4 describes the metrics we propose for measuring the performance of revisitation algorithms. In Chapter 5, we perform experiments on a dataset extracted from `avsforum.com`, and show that our models perform better than an average revisitation baseline. Chapter 6 then discusses our contributions, and possible avenues of future work.

Chapter 2

Related Work

In order to devise such an algorithm, we need to predict how often any user may update a page. Some work has been done to try to predict how often page content is updated, with the aim of scheduling download times in order to keep a local database fresh.

2.1 Refresh policies for incremental crawlers

We first discuss the *timeliness* of our crawler to maintain the freshness of the local database, which refers to how new the extracted information is. Web crawlers can be used to crawl sites for user comments for later post-processing. Web crawlers which maintain the freshness of a database of crawled content are known as incremental crawlers. Two trade-offs these crawlers face cited by Yang2009) are *completeness* and *timeliness*. *Completeness* refers to the extent which the crawler fetches all the pages, without missing any pages. *Timeliness* refers to the efficiency with which the crawler discovers and downloads newly-created content. We focus mainly on timeliness in this project, as we believe that timely updates of active threads are more important than complete archival of all threads in the forum site.

Many such works have used the Poisson distribution to model page updates. Coffman1997) analysed the theoretical aspects of doing this, showing that if the page change process is governed by a Poisson process $\frac{\lambda^k e^{-\lambda\mu}}{k!}$, then accessing the page at intervals proportional to λ is optimal.

Cho and Garcia-Molina trace the change history of 720,000 web pages collected over four

months, and showed empirically that the Poisson process model closely matches the update processes found in web pages (Cho1999). They then proposed different revisiting or refresh policies (Cho2003Garcia-molina2003) that attempt to maintain the freshness of the database.

The Poisson distribution were also used in Tan2007) and Wolf2002). However, the Poisson distribution is memoryless, and in experimental results due to Brian2000), the behaviour of site updates are not. Moreover, these studies were not performed specifically on online threads, where the behaviour of page updates differs from static pages.

Yang2009), attempted to resolve this by using the list structure of forum sites to infer a sitemap. With this, they reconstruct the full thread, and then use a linear-regression model to predict when the next update to the thread will arrive.

Forums have a logical, hierarchical structure in their layout, which typically alerts the user to thread updates by putting threads with new replies at the top of the thread index. Yang's work exploits this as well as their linear model to achieve a predicton of when to retrieve the pages. However this design pattern is not applied universally; comments on blog sites or e-commerce sites about products do not conform to this pattern. The lack of such information may result in a poorer estimate, or no estimate at all.

The above works all try to estimate the arrival of the next update (comment), but do not leverage an obvious source of information, which is the content of the posts themselves. Our perspective is that the available thread content can be used to provide a better estimation for predicting page updates.

Next, we look at some of the related work pertaining to thread content.

2.2 Thread content analysis

While there is little existing work using content to predict page updates, we review existing work related to analysing thread-based pages. We think such work will aid our efforts in content-based update prediction.

Wang2011) find links between forum posts using lexical chaining. They proposed a method to link posts using the tokens in the posts called $Chainer_{SV}$. While they analyse the contents

of individual posts, the paper does not make any prediction with regards to newer posts. The methods used to produce a numeric similarities between posts may be used as a feature to describe a thread in its current state, but incorporating this into our model is non-trivial.

There has been some work done recently in predicting events in social media, and in particular, tweets. Wang) dealt with predicting the retweetability of tweets using content. They applied two levels of classification, the first level categorising tweets into 6 different types: Opinion, Update, Interaction, Fact, Deals and Others. This was done using similar techniques as Sriram2010) and Naaman2010). The Opinion and Update categories are then further categorised into another three and two sub-categories each. The authors performed this categorisation using labeled Latent Dirichlet Allocation.

2.3 Conclusion

The state of current work related to revisitation policies mainly use estimations of previous update intervals to predict future update times. Analysis of user-generated content also do not tackle the problem of predicting when new content is created or published. These are the issues we will tackle with our work.

We aim to use the existing content available in the thread to train models for predicting when future posts will arrive. In the next chapter, we take a look at the various methods we propose for tackling this problem of revisitation.