# Predicting Web 2.0 Thread Updates

Shawn Tan

# Table of Contents

1. Introduction
2. Related work
3. Preliminary Work
4. Work Plan

# Motivation

- Many sites with thread-based discussion features
- Users post product reviews, feedback

Obtaining such up-to-date information may be vital to companies.

# Crawling forums: The Naive way

One way of keeping the database *fresh*, is to download pages at a frequent rate.
However, forum sites are too large, with too many threads, incurring high bandwidth costs.

# Crawling forums: Estimating Future posts

Attempt to estimate future posts by learning from intervals between past posts.

# Our approach

Use the content as well to attempt to make a better prediction.

- Technical forum discussions
- Flame wars (e.g. Vim vs. Emacs)

Ideally, an incremental crawler of such user-generated content should be able to maintain a fresh content.
However this

1. incur excessive costs when downloading un-updated pages
2. raise the possibility of the web master blocking the requester's IP address.

# Table of Contents

1. Introduction
2. Related work
3. Preliminary Work
4. Work Plan

# Refresh policies for incremental crawlers

Many works have used the Poisson distribution to model page updates.

1. Coffman et. al. 1997 analysed the theoretical aspects.
2. Cho and Garcia-Molina trace the change history of 720,000 web pages collected over 4 months.
    1. Showed empirically that the Poisson process model closely matches the update processes found in web pages (Cho et. al. 1999)
    2. Proposed different revisiting or refresh policies (Cho et. al. 2003, Garcia-molina et. al. 2003)
3. Also used in Tan et. al. 2007 and Wolf et. al. 2002.

# Problems with Poisson

The Poisson distribution is memoryless, and in experimental results due to Brewington and Cybenko 2000, the behaviour of site updates are not.

# Using Site-level Knowledge

Yang et. al. 2009, attempted to resolve this by

1. Using the list structure of forum sites to infer a sitemap.
2. Use a linear-regression model to predict when the next update to the thread will arrive.

# Table of Contents

1. Introduction
2. Related work
3. Preliminary Work
4. Work Plan

# Extracting Data

Collecting data from http://forum.hardwarezone.com.sg as our data set.

- User
- Timestamp
- Message body

Currently in raw HTML format, need to do preprocessing.

# Preliminary experiment

We picked a few threads (more than 3 days):

1. Computed time difference between posts $\Delta t$
2. Sort posts by time difference
3. Use median of $\Delta t$ as splitting point
4. Tran a Naive Bayes classifier to classify posts into 2 categories:
    - $\Delta t > 6$
    - $\Delta t \leq 6$

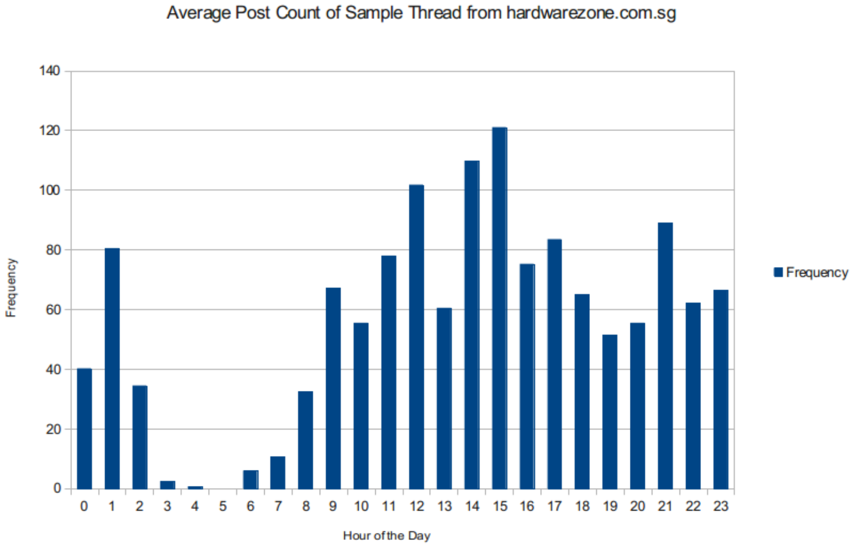
# Results

| Class | Precision | Recall | $F_1$ |
|---|---|---|---|
| $\Delta t \leq 6$ | 0.657 | 0.816 | 0.728 |
| $\Delta t > 6$ | 0.682 | 0.483 | 0.565 |

Table: Naive Bayes classification results

1. 10-fold cross validation for results
2. Low recall value for $\Delta t > 6$

# User posting frequency



# Evaluation metric

To evaluate *timeliness* of our algorithm we use the metric used in Yang et. al. 2009

$$T = \frac{1}{N}\sum_{i=1}^{N} \Delta t_i$$

# Observations

- Content has some type of relationship with the update rates
- Thread update rates also dependent on users’ sleep cycle

# Thread States

- Threads governed by probabilistic state machine
- Each state has an associated update rate, and set of observations (textual)

# Hidden Markov Model

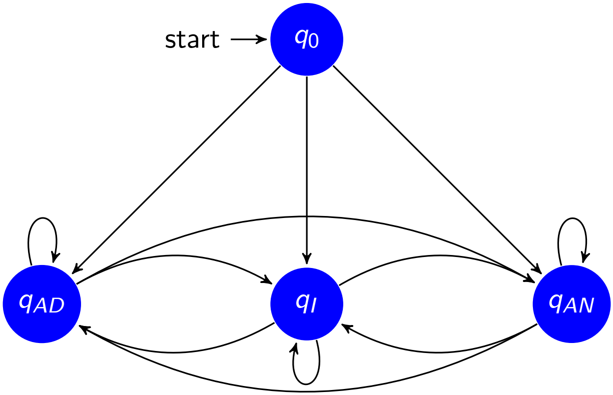


Figure: Modelling of a thread as a probabilistic automaton

# State observations

In the case of our thread content, possible observations include:

- Average length of a post
- Word frequencies
- Time between posts

# Table of Contents

1 Introduction

2 Related work

3 Preliminary Work

4 Work Plan

# Schedule

| Start date | End Date | Activity |
|---|---|---|
| 2012-05-07 | 2012-05-11 | Implement feature extraction from threads |
| 2012-05-14 | 2012-05-18 | Collect more data from other forums |
| 2012-05-21 | 2012-05-25 | Implement baseline using classification algorithm (SVM) |
| 2012-05-28 | 2012-06-01 | Implement linear regression from Yang et. Al. |
| 2012-06-04 | 2012-06-08 | Implement and test HMM |
| 2012-06-11 | 2012-06-15 | Implement and test HMM |
| 2012-06-18 | 2012-06-22 | Implement and test HMM |
| 2012-06-25 | 2012-06-29 | Implement and test HMM |
| 2012-07-02 | 2012-07-06 | Implement and test HMM |
| 2012-07-09 | 2012-07-13 | Evaluation |
| 2012-07-16 | 2012-07-20 | Evaluation |
| 2012-07-23 | 2012-07-27 | Evaluation |
| 2012-07-30 | 2012-08-03 | Evaluation |