Indexing Weblogs One Post at a Time

Natalie Glance
Intelliseek Applied Research Center
nglance@intelliseek.com

Abstract

In order to perform analysis over weblogs, we must first identify the appropriate unit of a weblog that corresponds to a document. We argue in the paper that, for weblogs, the correct unit is the weblog post. A weblog post is a structured document with the following fields: date, timestamp, title, content, permalink and author. We present our approach for segmenting weblogs into posts, which breaks down into several steps: (1) automatic feed discovery; (2) feed-guided segmentation, using the weblog feed and HTML; and (3) model-based weblog segementation.

Introduction

Web search engines such as Google, Yahoo and MSN Search index the entire content of a web page typically every few days. However, for weblogs, users want to be able to search over individual posts, and in near real-time. Weblog search portals such as Technorati, Feedster, PubSub and BlogPulse have gained in popularity over the past year and a half, as people begin to turn to weblogs to get up-to-the-minute breaking news and to get fresh angles on news stories. Major search engines are joining the fray: Google released a blog-only search engine end of September 2005 and a Yahoo blog search engine release is rumored to be imminent.

In addition, marketers have awakened to the possibility of mining consumer sentiment from weblogs. In order to produce accurate analytics, it is first necessary to be to identify individual weblog posts. Examples of consumer sentiment analytics are: the buzz surrounding a product (# mentions); # of links to a company website; trends in #mentions/#links; ratio of positive vs. negative mentions (Glance *et al.* 2005).

Researchers as well are turning to blogs to gauge opinion and community structure. For example, Adamic and Glance recently analyzed the linking behavior of political bloggers during the 2004 U.S. Presidential Election and found that conservative bloggers link to each other more frequently and in a denser pattern than liberal bloggers (2005). Marlow has studied the structure and authority in weblogs using inter-post citation counts (2004). Adar et. al. have explored how memes thread through the blogsphere from post to post (2004). The Global Attention Profiles project[1] tracks the attention that bloggers pay to different nations of the world, in comparison with selected mainstream media outlets. More recently, Gruhl et. al. reported that online blog postings can successfully predict spikes in the sales rank of books (2005). Building on these results, Mishne et. al. applied sentiment analysis to the context around movie mentions in weblog posts and found that positive mentions of movies correlate better to movie sales data than simple buzz counts in the pre-release time period (2006).

[1]http://h2odev.law.harvard.edu/ezuckerman/

To enable sophisticated analytics over weblogs, a blog search engine requires an indexing mechanism that indexes a weblog one post at time, as opposed to one HTML page at a time. In order to index blogs one post at a time, the indexing system must be able to segment the weblog HTML into individual posts and extract meta-data associated with the posts, such as the posting date, title, permalink and author.

In this paper, we describe our approach to segmenting weblogs into individual posts using a combination of weblog feeds (RSS, Atom) and model-based wrapper segmentation.

Defining and Modelling Weblogs

A weblog is commonly defined as a web page with a set of dated entries, in reverse chronological order, maintained by its writer via a weblog publishing software tool. In this paper, we define each entry as a set of one or more timestamped posts: an author may typically post several times a day. This is a matter a style: some authors post at most once a day in an all-inclusive entry. Others prefer to micro-post: each published item is a separate post in the day's entry.

Regardless, each post generally consists of: a date, a title, the body of the post, a permalink to the post, an author, and one or more categorizations. Often, the weblog home page will contain extraneous content on the page, such as a header, a footer and sidebars. However, the main content is a sequence of entries ordered in reverse chronological order, with each entry consisting of sequence of posts, also in reverse chronological order.

A weblog can be described formally as follows:

Weblog: Entry+

Entry: Date Post+

Post: Title? Content Permalink? Author? Timestamp? Link to comments? Categories*

The ordering of the sub-elements for the Entry elements and the Post elements is not standardized across weblogs, although it is assumed to be fixed within a weblog. Also, the model assumes that the entry dates are monotonically decreasing.

Weblog syndication

Many weblog publishing software tools also publish a feed in association with the weblog. The feed is updated whenever a new item is posted to the weblog. The feed is a “pull” mechanism, just as the weblog page is - the feed must be accessed in order to find out if it has been updated. However, feeds are designed to be read via a feed reader/aggregator (such as Bloglines, NewsGator, etc. or via an extension to a mail reader), which polls the feed on the behalf of the user(s). Thus, the end user who reads feeds via a feed reader experiences weblogs as a “push” phenomena: the newly published weblog posts are pushed to the user’s screen.

Some weblog software tools have provided customization of the weblog’s feed: the publication of the feed can be turned on/off; the feed can be updated whenever a new item is posted/modified; the feed can be full content/partial content. Full vs. partial content is an important distinction. We define a full content feed as a feed that publishes the entire content of the post as viewable on the front page of the weblog. We define a partial content feed as a feed that publishes a summary of the post content available via the weblog.

With respect to feed publication, weblog software tools fall into three categories: (1) automatic generation of feeds (partial or full); (2) customized generation of feeds; or (3) no feed generation capability. In the last case, some tech-savvy bloggers will use custom software to create a feed and associate it with their weblog, or turn to a third-party feed generator to host a feed for the weblog (e.g., FeedBurner: http://www.feedburner.com/).

Segmenting weblogs into posts

This section describes our approach for segmenting weblogs into posts. It would be prohibitively costly to manually create individual wrappers for each weblog. However, weblogs tend to conform to a common model, as described in Section . Thus, we have focused on developing an approach that generalizes well over all weblogs.

If a full content feed is available for a weblog, then the task of extracting posts from the weblog is the straightforward mapping of the XML format to an internal format. If a partial content feed exists for a weblog, then we use the partial content to guide the extraction process. If no partial content feed exists for a weblog, then we apply a model-based approach to extracting posts from the weblog page, taking advantage of regularities more or less common to most weblogs. Our work on model-based segmentation is similar to that of Nanno et. al. (2004).

Here is the process flow for our segmentation algorithm:

1. Discover set of feeds for the weblog.
2. Identify if feed is full content or partial content.
3. If feed is full content, we are done.
4. If feed items have dates & partial content, construct skeletal posts and fill in content using weblog page.
5. Otherwise, apply model-based weblog segmentation.

Feed discovery

The first step consists of discovering the feed(s) for the weblog. If the weblog update was collected from a ping server relaying extended pings, and if the accepted ping includes the feed URL for the weblog, then we are done. Alternatively, if the weblog is hosted by a weblog host which publishes full content feeds for all weblogs, then we need only map the weblog URL to the feed URL.

Otherwise, the next step in discovering the feed(s) for a weblog is to use RSS auto-discovery[2]. RSS auto-discovery is an agreed-upon standard for specifying the location(s) of a weblog’s feed(s) as meta-data in the HTML for the weblog home page.

If RSS auto-discovery fails to find a set of feeds for the weblog, the next step is to search for links to feeds from body of the weblog. First, all hyperlinks are extracted from the weblog. Next, the set of extracted hyperlinks are filtered using a classifier to identify which one(s) belong to the set of feeds for the weblogs. Currently, we use a set of heuristics to identify the feed(s) for a weblog from the extracted hyperlinks.

Full content vs. partial content feeds

The multiple XML standards for weblog feeds (several version s of RSS and Atom), all satisfy the following minimal conditions:

- The feed has the following top-level fields: **weblog url**, **weblog title**
- The feed consists of a set of items (which for weblogs, correspond to posts). Each item may have the following fields: **date-posted**, **permalink**, **post title**, **author**, **content**, **description**

Our feed finder considers an item in the feed valid if it contains, at minimum, the following fields: **date-posted** AND (**content** OR **description**). If no items are valid, the feed is rejected and weblog segmentation falls back upon model-based weblog segmentation (aka screen scraping).

The actual names of the fields depend on the feed standard being used. For example, for RSS v0.91, **date-posted** maps onto the XPath **/item/title**; **content** maps onto the XPath **/item/description**; and **description** maps onto the XPath **/item/description**. (There is no separate content field in the RSS v0.91 specification.)

Typically, the **description** field is used to provide a summary of the post (usually the first few lines) while the **content** field is used to provide either the full content of the post or a summary. Some feeds contain both, in which case, typically, the **description** contains the summary and the **content** contains the full post.

The feed classifier, which classifies the feed as full content or partial content, takes as input features of the **content** and

[2]http://diveintomark.org/archives/2002/05/30/rss_autodiscovery

description text, such as: presence/absence of HTML tags; % posts ending in ellipses; and type of feed. Based on these features, it uses heuristics decides whether or not the items in the feed are full content. Other features could be added, such as the variance in the length of text, etc.

If the feed is classified as full content, then we map the data found into the feed into our own internal representation for weblog posts.

If the feed is not full content, then we create skeletal posts from the data in the field. For each post, we fill in the following data: **weblog url**; **date-posted**; **partial content**; **post title** (if found); **post author** (if found); and **permalink** (if found).

Feed-guided weblog segmentation

The next step is to fill in the skeletal posts constructed from the feed by using the content of the weblog page itself. Missing from the skeletal posts is the full content of the post. To find the full content, first, the partial content is processed to remove summarization artificats (e.g., ending ellipsis). In addition, advertisements included in the feed (for example FeedBurner ads) are removed. Then, we search for the partial content in the weblog. If the partial content is not found, then we punt on the post. If we end up punting on all posts, then we will fall back on model-based segmentation.

If the partial content matches text on the weblog home page, then we find the enclosing node for the matching text in the tidied XHTML for the weblog page. The text inside the enclosing node is then used as the content for the post. If enclosing nodes for successive posts overlap, then we throw an error indicating that feed-guided segmentation has failed for the weblog, and, again, fall back on model-based segmentation.

Model-based weblog segmentation

Model-based weblog segmentation assumes that weblogs can be modelled as described in Section . Our approach then starts from a simplification of that model: (**date** ([**title**] **content**)+)+. This model which assumes that dates appear first. This means that if we are able to extract the weblog entry dates, then we can use the dates as markers for the entries. Of course, a weblog page may have many other dates apart from the dates marking the entries: dates in the content of the posts; dates in the sidebars or in other non-weblog content included in the HTML page. However, as weblogs are produced by weblog software, we can expect certain regularities in the underlying DOM of the generated HTML. In particular, we expect that the relative XPaths of the weblog entry dates to be identical. In practice we've found that the relative XPaths of the entry dates are identical if we ignore certain elements in the XPath: /align/ and repeating /font/s.

The first step in our model-based segmentation algorithm consists of extracting all the dates from the tidied XHTML for the weblog page using a date extractor. The dates are sorted into ordered lists, one list for each unique relative XPath. The order within the list corresponds to the ordering of the dates with the DOM for the weblog page.

We then filter the lists according to a set of heuristics in order to identify which list corresponds to the actual weblog entry dates:

1. Keep only lists whose dates all belong to the current year and/or the past year.
2. Keep only non-singleton date lists.
3. Keep only lists whose dates conform to a similar format (e.g. MM/dd/YYYY).
4. Keep only lists whose dates decrease monotonically.
5. Keep only lists with most recent dates (but not in the future).
6. Keep only lists with longest date string representation.
7. Keep only lists with the greatest number of dates.
8. Keep only first list.

One might think that after step 5 in the filtering process, we would be left with at most one list of dates. In practice, this is not the case, because weblogs often have a sidebar with a dated list of recent posts which corresponds exactly the full set of posts in the main part of the weblog. The last few filtering steps help correctly identify the weblog entry dates as opposed to the dates in the sidebar.

If we fail to find a conforming list of dates, then model-based segmentation fails. There are some known cases where our approach fails: when only one entry appears on the home page of the weblog; or when weblog software for some reason generates irregular XPaths for the dates and/or content. But in many cases, segmentation fails when the HTML page in question is not actually a weblog. Thus, our model-based segmentation algorithm has the additional functionality of serving as a classifier that identifies whether or not an HTML page is indeed a weblog.

Once we have identified the entry dates for the weblog, model-based segmentation proceeds as follows:

1. Segment weblog into entries, using dates as markers.
2. Segment each weblog entry into posts using post titles markers.
3. For each post, identify permalink and author.

In step 1, we assume that all DOM nodes between subsequent entry dates form the weblog entry associated with the earlier date. The main difficulty is identifying the end of the last post. For this we use a set of heuristics to identify the end of the blog entry by looking for the start of boilerplate weblog end template. Example end markers include: the start of a sidebar or a copyright notice.

In step 2, we attempt to use post titles to demarcate boundaries between posts for an entry. First, we iterate over the nodes of the entry searching for a node that matches one of our conditions for being a title node. These conditions include: class attribute of the node equals 'title' or 'subtitle' or 'blogpost', etc. Once we have found the first matching title, we then assume that all subsequent post titles will have the same relative XPath. Again, we assume that all DOM nodes between subsequent title nodes are associated with the earlier title.

If we are unable to find titles, then we treat the entire entry as a single post. In fact, we have found that the majority of bloggers do not post more than once per day.

Segmentation method	% of weblogs
Full content feed	78%
Feed-guided segmentation	11%
Model-based segmentation	11%

Table 1: Segmentation statistics for 4/13/2005

The final post-processing step identifies the permalink and author from the content of each extracted post using common patterns for permalinks and author signatures. To find authors, we look for patterns like "posted by." To find permalinks, we look for hrefs in the post content that match, for example, "comment" or "archive." Some patterns are given higher priority than others for matching against permalinks.

A weakness of our current implementation of model-based wrapper segmentation is that it assumes that the date field comes first in a weblog entry. In fact, while most blogs exhibit the pattern **date** ([**title**] **content**)+, others use (**title date content**)+ or even ([**title**] **content date**)+. Our approach is still able to segment blogs exhibiting these less common patterns, although the segmentation associates the date with the incorrect content. That is, if we have a sequence of N posts (post 1 through post N), the date for post 1 will be associated with the content of post 2 and so on. In addition, we will fail to extract the content of post 1. We call this error a parity error.

Segmentation statistics

We have implemented weblog segmentation as part of the BlogPulse weblog post collection, indexing and search system.

In tests of the model-based segmentation algorithm, we have found that the precision of this algorithm is about 90% – that is about 90% of extracted posts have date, title and content fields that correspond to those of actual posts on the weblogs. The recall is approximately 70% – that is, we are able to extract posts from about 70% of true weblogs.

Table 1 shows the statistics for our segmentation process, the percentage of weblogs segmented using: (1) full content feeds (78%); (2) feed-guided segmentation (11%); or (3) model-based segmentation (11%).

Conclusion

In this paper, we have described a process for segmenting weblogs into posts. In developing this process, we have focused on optimizing the generalizability of our segmentation algorithms while maintaining high coverage. That is, the goal was to have a segmentation process that would work with a large majority of blogs without require specialized handling for individual weblogs.

We have implemented our segmentation algorithm as part of the weblog post collection subsytem of BlogPulse. This enables BlogPulse to provide search over individual blog posts. Furthermore, the corpus of dated weblog posts serves as a data set for tracking trends over time, and for analyzing how memes spread through the blogosphere.

Acknowledgements

The author has benefited greatly from discussions with Matt Hurst.

References

Adamic, L., and Glance, N. 2005. The political blogosphere and the 2004 u.s. election: Divided they blog. In *Proceedings WWW-2005 2nd Annual Workshop on the Weblogging Ecosystem*.

Adar, E.; Zhang, L.; Adamic, L. A.; and Lukose, R. M. 2004. Implicit structure and the dynamics of blogspace. In *Proceedings WWW-2004 Workshop on the Weblogging Ecosystem*.

Glance, N.; Hurst, M.; Nigam, K.; Siegler, M.; Stockton, R.; and Tomokiyo, T. 2005. Analyzing online discussion for marketing intelligence. In *Proceedings WWW-2005*.

Gruhl, D.; Guha, R.; Kumar, R.; Novak, J.; and Tomkins, A. 2005. The predictive power of online chatter. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 78–87. New York, NY, USA: ACM Press.

Marlow, C. 2004. Audience, structure and authority in the weblog community. In *International Communication Association Conference*.

Mishne, G., and Glance, N. 2006. Predicting movie sales from blogger sentiment. Submitted to AAAI Spring 2006 Symposium on Computational Approaches to Analyzing Weblogs.

Nanno, T. 2004. Automatic collection and monitoring of japanese weblogs. In *Proceedings WWW-2004 Workshop on the Weblogging Ecosystem*.