Design and Implementation of Distributed Crawler System for Opinion Mining

Yunqi Gao, Chunlin Peng
Research Institute of Elect Science and Technology of UESTC
University of Electronic Science and Technology of China
Chengdu, China
{mopyfish, peng_chunlin}@163.com

***Abstract*—With the development of Internet, Network public opinion has been serving an import role in reflection of social public opinion. As there are a large number of websites and forums on the Internet, we need a powerful crawler system which can meet the demands of opinion mining. However, common crawler systems concern more about ranking and recommendation algorithms, which is less important in opinion mining. In this article, we introduced the design and implementation of a distributed crawler system for opinion mining. We also introduced some extra parameters such as keywords count and published time into the ranking and refreshing strategies. Experimental results demonstrate that the system can well support different sites, and the improved strategies can greatly enhance the crawling and monitoring efficiency.**

Keywords-distributed system; crawler; public opinion; PageRank;

I. INTRODUCTION

As the continuous development of Internet, it has become the fourth media following the traditional media, and it has made comprehensive and profound impacts on people's daily life. In Web 2.0 era, platforms such as forum, blog and microblog provide a wider channel to express people's views, and traditional news websites are improving their interactive features, so that people can participate in the discussion of hot events [1]. For government, in order to master the latest network public opinion, an efficient and accurate opinion monitoring system which can cover major websites has become and actual demand.

Distributed crawler system is the basis for opinion monitoring system, and its speed and support of websites will directly affect the subsequent analysis process [2]. Different from general crawlers, crawler using in opinion monitoring systems are well targeted. In default settings, the crawler will only focus on news sites, forums, blogs and microblogs where there exist frequent interactions, which requires the crawler can not only support the dynamic pages, but also can adjust its crawling frequency according to the analysis result.

In this article, we purposed the crawler system Yakamoz for the field of opinion monitoring. It support the crawling of major websites (including the comment of the news), and can adjust its crawling strategy in realtime according to the analysis result.

II. INTRODUCTION OF DISTRIBUTED CRAWLER

Web crawler is one of the most important modules for search engines as well as related data mining systems [3]. Generally, the crawler downloads html source code of the web pages, and uses regular expression to find all URLs in the downloaded pages, and use these URLs as seeds for next turn of download. To collect information efficiently, we often use distributed crawler to do parallelized crawling.

When we need to extract specific information from the page, we can integrate extra modules into the crawler to make it a vertical crawler [4]. For examples, we can build templates (no matter by regular expression or by DOM parsing) to extract certain fields, or use Javascript engines to parse dynamic pages [5].

Besides the architecture of the system, there also can be improvements in related algorithms, including traditional PageRank algorithm [6]. For example, we can rank the pages by user's interest as well as relation of links [7].

III. DESGIN AND IMPLEMENTATION OF THE SYSTEM

Yakamoz is a distributed master-slave crawler system, which comprises of a central control node, a crawler manager (master node) with a backup node, and several crawler (slave) nodes. Besides, the system provides a series of interfaces, by which the crawler can be directly connected to the analyzing system, and get the result for feedback. The overall architecture of the system is shown in Fig. 1.

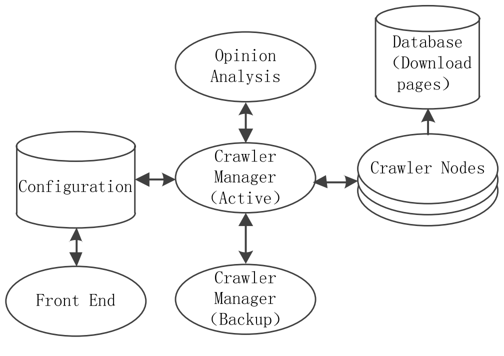

Figure 1. Architecture of the system

Besides the crawlers, the system provides a front end for management, by which users can configure the rules and strategies of the crawlers, as well as view the status. When integrated with public opinion analysis system, the crawler system can share the front end with the opinion system.

A. Crawler Manager

Crawler manager is the core of the whole system, including task and strategy parsing module, URL management module and task scheduling module. Its workflow is shown in Fig. 2.

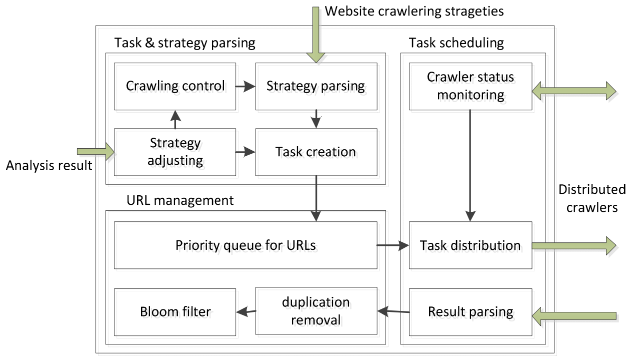

Figure 2. Workflow of crawler manager

1) Task and stragety parsing: In this system, each site corresponds to a strategy file, which describes the starting URLs (seeds), crawling interval, number of threads and the downloader (A module to download webpages) using for this site. In the startup period, the manager will check whether there are modifications, if any, it will synchronize the modifications to all slave nodes. Besides, it will merge the seeds of all files into the management module as the total task.

2) URL management: In the master node, we use a bloom filter to store crawled URLs. Considering the actual scale of deployment, the filter is deployed on master node, instead of on each of the slaves, so the master node can deploy the task to any idle node and don't need to consider the partitioning, which makes the load balance more effective. For URL storage, we use a five-level priority queue, and by default all URLs are treated as level 3. Besides, in each level of the queue, the URLs are grouped by their hash values, so that the crawler will visit different sites in one time. Considering the master node may crash, the queue and the filter will be stored in the database periodically, so when the master node fails, the backup master node will resume the status from the database, thus minimum the impact of the node failure.

3) Task scheduling: By default, the task scheduling module will split the task into fragments, each of which has a size of 100-200 URLs, and deploy them to different crawlers. At the same time, the module will maintain the status of each crawler and task. Every 60 seconds, the module will send a heartbeat packet to the crawlers to get their status. For a task, if it isn't submitted in certain time (by default URL numbers x 5 seconds), the module will fail this job and deploy it to another crawler. As each of the tasks has a unique number, so the module can check this number to avoid repeatedly submitting.

B. Crawler

The crawler is the performer of specific tasks, and consists of a crawler controller, a downloader factory and several working threads. The workflow is shown in Fig. 3.

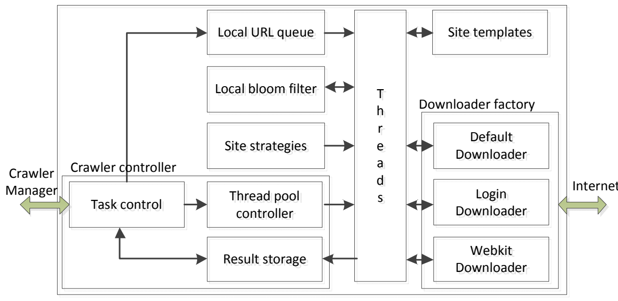

Figure 3. Workflow of crawler

1) Crawler controller: This module is to control the entire operation of the crawler, and in charge of the communication to the master node. Task control module is responsible for task parsing and result submitting, as well as timeout control. Thread pool control module is responsible for managing all working threads, and restart the corresponding thread if it fails. To reduce the load of the database, the temporary result will be first stored in the local storage and submitted to the database when the task is finished. If a task failed repeatedly, the controller will store it to the local disk for later analysis.

2) Downloader factory: To ensure the performance of multi-thread download, each thread can obtain the instances of all downloaders except the Webkit downloader considering its resources consumption. At present, there are three kinds of downloaders in the system.

- *Default downloader:* Fastest of all downloaders, generally used to download static webpages, also support user-define HTTP headers and proxies.
- *Login downloader:* Login downloader can store the usernames and passwords of the websites, and mainly used to visit microblogs and forums.
- *Webkit downloader:* Webkit downloader uses the Webkit browser kernel provided by QT to visit dynamic pages, and support AJAX calls, and can be used to download the comments of news or the homepage of microblog. Considering its resources consumption, there's only one instance in the system, and all threads call it serially.

IV. ALGORITHMS AND STATIGIES IN THE SYSTEM

A. Task scheduling strategy

Task scheduling strategy includes two parts: Task deploying and result processing. The description of task deploying is as follows:

a) Traverse crawler list *QC*, find a crawler C_{IDLE} which status is *IDLE*;